(19)日本国特許庁（ＪＰ）　(12) **公開特許公報(A)**　(11)特許出願公開番号

特開平8－314872

(43)公開日　平成８年(1996)11月29日

(51)Int.Cl.[6]		識別記号	庁内整理番号	ＦＩ		技術表示箇所
Ｇ０６Ｆ	15/16	３７０		Ｇ０６Ｆ 15/16	３７０Ｚ	
	9/46	３６０		9/46	３６０Ｆ	
	15/00	３９０	9364－5L	15/00	３９０	

審査請求　有　　請求項の数７　ＦＤ　(全 32 頁)

(21)出願番号　特願平7－138641

(22)出願日　平成７年(1995)５月12日

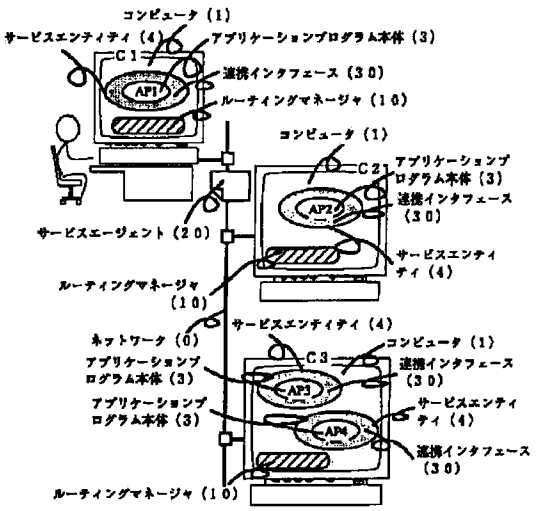
(71)出願人　000233538
日立東北ソフトウェア株式会社
宮城県仙台市青葉区一番町２丁目４番１号
(72)発明者　伊藤　俊明
宮城県仙台市青葉区一番町二丁目４番１号
日立東北ソフトウェア株式会社内
(72)発明者　樋地　正浩
宮城県仙台市青葉区一番町二丁目４番１号
日立東北ソフトウェア株式会社内
(72)発明者　岡崎　司
宮城県仙台市青葉区一番町二丁目４番１号
日立東北ソフトウェア株式会社内
(74)代理人　弁理士　須田　篤

最終頁に続く

(54)【発明の名称】　アプリケーションプログラム間連携処理方法

(57)【要約】
【目的】個々の利用者が自分自身の利用方法に適するアプリケーションプログラム間の連携処理を行うことを可能とする。
【構成】複数のアプリケーションプログラム３の利用手順とその各々の利用手続きを記述した作業指示書（サービスエージェント）２０を作成し、該作業指示書を複数のコンピュータ１へ順次転送する。各アプリケーションプログラム３には、作業指示書に記載された利用手続きに応じて当該アプリケーションプログラムとの連携をとる連携インタフェース３０を設けておき、各連携インタフェース３０は、作業指示書２０を受けたとき、該連携インタフェースが当該作業指示書に記載された自己のアプリケーションプログラムに関する利用手続きに応じて自己のアプリケーションプログラムを起動し、目的とする制御および／またはデータの授受を行う。

図１　アプリケーションプログラム間連携処理システム構成



1

【特許請求の範囲】

【請求項1】コンピュータネットワークに接続された複数のコンピュータ上で動作する複数のアプリケーションプログラム間の連携処理方法において、

前記複数のアプリケーションプログラムの利用順序と各アプリケーションプログラムの利用手続きを記述した作業指示書を作成し、

該作業指示書を、前記コンピュータネットワークを介して、目的のアプリケーションプログラムを有する複数のコンピュータへ順次転送し、

前記複数のアプリケーションプログラムの各々には、前記作業指示書に記載された当該アプリケーションプログラムの利用手続きに応じて当該アプリケーションプログラムとの連携をとる連携インタフェースを設けておき、

各連携インタフェースは、前記作業指示書を受けたとき、該連携インタフェースが当該作業指示書に記載された自己のアプリケーションプログラムに関する利用手続きに応じて自己のアプリケーションプログラムを起動し、目的とする制御および／またはデータの授受を行い、

必要なアプリケーションプログラムを有するコンピュータを作業指示書が一巡することにより前記複数のアプリケーションプログラムによる一連の作業を実行することを特徴とするアプリケーションプログラム間連携処理方法。

【請求項2】請求項1記載のアプリケーションプログラム間連携処理方法において、前記作業指示書に記載された利用手続きの構成要素として、当該作業指示書が利用する全てのアプリケーションプログラムで使用されるデータ項目の一覧を表す全データ項目、各アプリケーションプログラムへ与えるべきデータが格納されている出力データ項目、各アプリケーションプログラムの処理結果が格納される入力データ項目、前記出力データ項目に格納されているデータを当該アプリケーションプログラムへ出力する手続きを表すデータ出力手続き、前記入力データ項目ごとのデータの入力手続きを表すデータ入力手続き、該データ入力手続きを用いて入力されたデータを処理する入力データ処理手続き、当該アプリケーションプログラムの実行を制御する制御手続き、のうち少なくとも1つを含むことを特徴とするアプリケーションプログラム間連携処理方法。

【請求項3】請求項1または2記載のアプリケーションプログラム間連携処理方法において、各コンピュータ上に移動制御手段を有し、該移動制御手段が、自己のコンピュータ上の前記連携インタフェース及び他のコンピュータとの間の通信路を確保、管理し、該通信路を通して受信した前記作業指示書に記載された利用順序に従って当該作業指示書の移動を制御することを特徴とするアプリケーションプログラム間連携処理方法。

【請求項4】請求項2記載のアプリケーションプログラム間連携処理方法において、前記作業指示書の構成要素である制御手続きとして、該作業指示書を受信した前記連携インタフェースのアプリケーションプログラムを起動する起動手続き、該アプリケーションプログラムの実行を終了する終了手続き、該アプリケーションプログラムの有する1つ以上のコマンドを実行する実行手続き、それらを組み合せた制御スクリプトの少なくとも1つの手続きを有することを特徴とするアプリケーションプログラム間連携処理方法。

【請求項5】請求項2記載のアプリケーションプログラム間連携処理方法において、前記作業指示書の構成要素として、新たな作業指示書を生成するための生成手続きを有することを特徴とするアプリケーションプログラム間連携処理方法。

【請求項6】請求項1または2記載のアプリケーションプログラム間連携処理方法において、前記利用手続きは、プログラムおよび関数名のいずれかであり、関数名の場合には前記連携インタフェースにおいて当該関数名に対応するプログラムを取得することを特徴とするアプリケーションプログラム間連携処理方法。

【請求項7】請求項1，2，3，4，5または6記載のアプリケーションプログラム間連携処理方法において、前記作業指示書に記述された利用順序は、コンピュータ名と、そのコンピュータ上で動作するアプリケーションプログラムおよび連携インタフェースからなるサービスエンティティの名称との組を該サービスエンティティの利用の順序に従って指定したものであり、前記コンピュータ名を省略可能としたことを特徴とするアプリケーションプログラム間連携処理方法。

【発明の詳細な説明】

【0001】

【産業上の利用分野】本発明は、利用者が複数のアプリケーションプログラムを順次組み合わせながら利用して処理を行う分散処理システムに係り、必要に応じてその処理経過や処理結果、処理内容を他の作業を行っている各作業者との間でやり取りすることにより、お互いの作業を連携させながら進めていく一連の作業から構成される業務において、これらの業務を行う際に各部署や作業者が利用するさまざまなアプリケーションプログラムの間の処理を連携させ、さらに業務に応じて個々のアプリケーションプログラムの間の処理の連携手続きを容易に変更することを可能とするアプリケーションプログラム間の連携処理方法に関する。

【0002】特に、コンピュータネットワークに接続されたコンピュータ上で動作する各種アプリケーションプログラムにより構成される分散処理システムにおいて、利用者が行っていた複数のアプリケーションプログラムを組み合わせた操作手順や、各作業者に依頼して行っていたアプリケーションプログラムの操作やその結果の取得を自動化するとともに、一連の作業に必要なアプリケ

3

ーションプログラムの利用手順をシステム化することにより、各作業者の作業負荷を軽減し、作業の流れを円滑にするために、作業に応じて柔軟にアプリケーションプログラムを組み合わせ、連携して処理するための利用技術、及びこのような柔軟なアプリケーションプログラムの連携処理方法を実現するための分散処理システムに関する。

【０００３】

【従来の技術】分散処理システムは、個々の作業者が行うべき作業に用いられるコンピュータをネットワークで接続し、個々の作業者間で必要とするデータをやり取りするシステムである。このようなシステムでは、１つ１つの作業は個々の作業者がアプリケーションプログラムを用いて行う個人利用中心であり、各作業者の間で必要とするデータだけが共有され、各作業者の利用するアプリケーションプログラムで共有されたデータを利用できる。

【０００４】分散処理システム上でアプリケーションプログラムを連携して動作させるための仕組みに関しては、日経コンピュータ：「クライアント・サーバーの弱点 非同期、蓄積型通信が解消」1994.8.8. pp.159-168、日経コンピュータ：「分散アプリケーション同士の非同期連携を容易に実現する　メッセージ・ベースのミドルウェアが急増」1993.9.20. pp.67-74 に、RPC（Remote Procedure Call）、会話型、ファイル転送、メッセージ・ベースの４つの連携処理方式について論じられている。

【０００５】「RPC」は、アプリケーションプログラムの１つのモジュールを取りだし、そのモジュールを他のコンピュータで実行させる方式である。「会話型」は、複数のアプリケーションプログラムの間をリアルタイム通信路で接続し、そのリアルタイム通信路を通じてデータや処理要求をやり取りする方式である。会話型の方式を用いて、或る作業に利用するアプリケーションプログラムを処理の要求元（クライアント）と処理の実行元（サーバ）に分割しクライアント側アプリケーションプログラムとサーバ側アプリケーションプログラムの間で連携して処理を行えるようにしたものがクライアント・サーバ・システムである。これらの２つの方式では、受信側アプリケーションプログラムは常に動作し、送信側アプリケーションプログラムからの処理要求を待つ状態にあり、送信側アプリケーションプログラムからの処理要求を受け取るとそれを即座に処理し、処理結果を送信側アプリケーションプログラムに返すことにより、送信側アプリケーションプログラムと受信側アプリケーションプログラムの間で同期を取りながら処理を行う連携処理を実現できる。また、これらの方式では、連携処理を行うアプリケーションプログラムの間の連携手続きをアプリケーションプログラムを開発する際にあらかじめ定め、その手続きをプログラムとして記述しておく必要がある。

4

【０００６】「ファイル転送」は、連携させるアプリケーションプログラムの間でファイルを転送し、ファイルに記述されたデータを受け取った場合にアプリケーションプログラムがそのデータに対する処理を行う方式である。この方式では、ファイルの受信をトリガとしてアプリケーションプログラムを実行させることにより、非同期的なアプリケーションプログラム間の連携処理を実現できる。「メッセージ・ベース」は、送信側アプリケーションプログラムから送り出されたメッセージをメッセージ・ベースの処理を実現するプログラムが受け取り、メッセージに記述された受信側アプリケーションプログラムにそのメッセージを送り、メッセージを受け取った受信側アプリケーションプログラムではメッセージに対応した処理手続きを実行することにより、アプリケーションプログラム間の連携処理を実現する方式である。メッセージ・ベースの方式では、連携処理を行うアプリケーションプログラムの間であらかじめメッセージの形式を定め、そのメッセージを受け取った場合のアプリケーションプログラムの処理を記述する。

【０００７】これら４つのアプリケーションプログラム連携処理方式では、一連の作業から構成される業務を、個々の作業で用いられる複数のアプリケーションプログラムの間で処理を連携させることにより行うためには、連携のために必要な処理手続きを各アプリケーションプログラムごとに定め、プログラムとして記述しておく必要がある。

【０００８】特開平2-186737号公報に記載のプログラム間論理通信路制御方式では、ネットワークに接続されたコンピュータ上で動作するアプリケーションプログラムの間で、データを交換することにより、アプリケーションプログラム間の連携処理を実現する方式について述べられている。この方式では、アプリケーションプログラムの画面を介してユーザにより加えられた操作（データの変更、追加、削除など）のデータを論理的通信路を通じて同一のアプリケーションプログラム間でやり取りすることにより、そのアプリケーションプログラム間で画面の表示を共有することに基づくアプリケーションプログラム間の連携処理を実現している。

【０００９】これらのアプリケーションプログラム間の連携処理方式においては、連携処理に用いるデータやメッセージを他のコンピュータ上のアプリケーションプログラムに送る際には、送信先のコンピュータの名称、送信先コンピュータの物理的な位置または物理的名称のいずれか１つの方法でデータやメッセージの送信先を指定することにより、指定されたコンピュータ上のアプリケーションプログラムにデータやメッセージを送ることができる。

【００１０】最近では、個々の作業者による個人利用中心のシステムから複数の作業者の間にまたがる作業を支

5

援するシステムの研究・開発が盛んに行われている。これら複数の作業者に跨る作業を支援するシステムの種類とその支援する作業の内容に関しては、石井裕：「グループウェア技術の研究動向」情報処理学会論文誌 Vol.30 No.12 Dec.1989、石井裕：「コンピュータを用いたグループワーク支援の研究動向」コンピュータソフトウェア Vol.8 No.2 pp.14-26、C.A. Ellis, S.J. Gibbs and G.L.Rein：”Groupware：Some Issues and Experiences”,CACM Jan.1991 Vol.34, No.1 pp.38-58.、に国内外で研究、開発が行われている複数の人々の間に跨って行われる協同作業を支援するシステムが論じられている。

【００１１】これらの協同作業を支援するシステムは、各作業者が同一のアプリケーションプログラムを利用して同一の内容の作業を協同して行うこと（協同文書作成など）を支援するシステムであり、各作業者の間でアプリケーションプログラムの処理結果を表示し、処理結果に対して何らかの操作を加えるユーザインタフェース画面を共有する機能を提供する。

【００１２】

【発明が解決しようとする課題】上記従来技術は、或る特定のアプリケーションプログラム間でやり取りするデータやメッセージをあらかじめ定め、その定めたデータやメッセージに対応する処理プログラムをそれぞれのアプリケーションプログラムに記述し、このあらかじめ定められたデータやメッセージをアプリケーションプログラム間でやり取りすることにより、アプリケーションプログラム間の連携処理を行う。そのため、アプリケーションプログラム間の連携処理の手続きを変更することはできない。あるいは、変更できたとしても、変更に関連する全てのアプリケーションプログラム中の処理プログラムを変更する必要がある。その結果、作業内容や目的に応じて容易にアプリケーションプログラム間の連携処理手続きを変更することは非常に困難となっている。

【００１３】さらにアプリケーションプログラム間の連携処理手続きを変更するためには、その変更対象となる全てのアプリケーションプログラムの実行をいったん終了させる必要がある。分散処理システム上ではさまざまな利用者が各々の作業を行うためにアプリケーションプログラムを使用しており、変更対象となる全てのアプリケーションプログラムの実行をいったん終了させることは、変更対象となるアプリケーションプログラムの数が増加するに従い、困難さを増してくることになる。

【００１４】また、アプリケーションプログラム間で連携処理を行うためにやり取りされるものは、データもしくはメッセージであるため、それらデータやメッセージを受け取った際の処理手続きは、それを受け取った受信側アプリケーションプログラムにより決められており、送信側アプリケーションプログラムが受信側アプリケーションプログラムの処理手続きを変更することは不可能

6

であった。また、受信側アプリケーションプログラムの提供する複数の機能を組み合わせた一連の処理手続き、すなわち受信側アプリケーションプログラムで一度に実行したい複数のコマンドの組み合わせを実行すること、を実現するためには、毎回、同期を取りながら必要な処理手続きを実行するためのメッセージをアプリケーションプログラムに送り、該当する手続きを実行し、その実行結果を送信側アプリケーションプログラムで組み合わせるか、あらかじめ受信側アプリケーションプログラムに複数のコマンドを実行するための特別なメッセージをプログラムとして記述しておく必要があり、開発効率が低下する。

【００１５】このようにアプリケーションプログラム間の連携処理手続きをあらかじめ記述しておく必要があるため、個々の利用者が自分自身の利用方法に適するアプリケーションプログラム間の連携処理手続きを記述することは困難であり、システムにより提供されるアプリケーションプログラム間の連携処理手続きを利用する以外には、アプリケーションプログラム間を連携して処理させることは困難であった。

【００１６】さらにこれらのデータやメッセージをアプリケーションプログラムに送るためには、そのアプリケーションプログラムの動作しているコンピュータをコンピュータの名称、またはIP アドレス等のコンピュータを一意に識別するための手段で明示的に指定する必要があり、或る処理を行うアプリケーションプログラムの動作するコンピュータを何等かの理由で変更した場合には、該当するアプリケーションプログラムを利用しているアプリケーションプログラム連携処理手続きの中のコンピュータを指定している部分を変更しなければならない。そのため、分散処理システムを構成するコンピュータやネットワーク、アプリケーションプログラムの変更に対する柔軟性に欠ける。

【００１７】本発明の目的は、個々の利用者が自分自身の利用方法に適するアプリケーションプログラム間の連携処理を行うことを可能とするアプリケーションプログラム間連携処理方法を提供することにある。

【００１８】本発明の他の目的は、データやメッセージに加え、アプリケーションプログラム間の連携処理手続き、データやメッセージを受け取ることにより、受信側アプリケーションプログラムで行われる処理手続き、受信側アプリケーションプログラムの提供する複数の機能を組み合わせた一連の処理手続きをアプリケーションプログラム間で送受信し、受信したこれら手続きを実行できる連携インタフェースをアプリケーションプログラムに付与することにより、アプリケーションプログラム間の連携処理手続きの変更を柔軟、かつ開発効率を低下させることなく行うことのできるアプリケーションプログラム間連携処理方法を提供することにある。

【００１９】

7

【課題を解決するための手段】上記目的を達成するために、本発明は、コンピュータネットワークに接続された複数のコンピュータ上で動作する複数のアプリケーションプログラム間の連携処理方法において、前記複数のアプリケーションプログラムの利用順序と各アプリケーションプログラムの利用手続きを記述した作業指示書を作成し、該作業指示書を、前記コンピュータネットワークを介して、目的のアプリケーションプログラムを有する複数のコンピュータへ順次転送し、前記複数のアプリケーションプログラムの各々には、前記作業指示書に記載された当該アプリケーションプログラムの利用手続きに応じて当該アプリケーションプログラムとの連携をとる連携インタフェースを設けておき、各連携インタフェースは、前記作業指示書を受けたとき、該連携インタフェースが当該作業指示書に記載された自己のアプリケーションプログラムに関する利用手続きに応じて自己のアプリケーションプログラムを起動し、目的とする制御および／またはデータの授受を行い、必要なアプリケーションプログラムを有するコンピュータを作業指示書が一巡することにより前記複数のアプリケーションプログラムによる一連の作業を実行するようにしたものである。

【００２０】このアプリケーションプログラム間連携処理方法において、前記作業指示書に記載された利用手続きの構成要素として、例えば、当該作業指示書が利用する全てのアプリケーションプログラムで使用されるデータ項目の一覧を表す全データ項目、各アプリケーションプログラムへ与えるべきデータが格納されている出力データ項目、各アプリケーションプログラムの処理結果が格納される入力データ項目、前記出力データ項目に格納されているデータを当該アプリケーションプログラムへ出力する手続きを表すデータ出力手続き、前記入力データ項目ごとのデータの入力手続きを表すデータ入力手続き、該データ入力手続きを用いて入力されたデータを処理する入力データ処理手続き、当該アプリケーションプログラムの実行を制御する制御手続き、のうち少なくとも１つを含む。

【００２１】好ましくは、各コンピュータ上に移動制御手段を有し、該移動制御手段が、自己のコンピュータ上の前記連携インタフェース及び他のコンピュータとの間の通信路を確保、管理し、該通信路を通して受信した前記作業指示書に記載された利用順序に従って当該作業指示書の移動を制御する。

【００２２】前記作業指示書の構成要素である制御手続きとして、例えば、該作業指示書を受信した前記連携インタフェースのアプリケーションプログラムを起動する起動手続き、該アプリケーションプログラムの実行を終了する終了手続き、該アプリケーションプログラムの有する１つ以上のコマンドを実行する実行手続き、それらを組み合せた制御スクリプトの少なくとも１つの手続きを有する。

8

【００２３】前記作業指示書の構成要素として、新たな作業指示書を生成するための生成手続きを有してもよい。前記利用手続きは、例えば、プログラムおよび関数名のいずれかであり、関数名の場合には前記連携インタフェースにおいて当該関数名に対応するプログラムを取得する。

【００２４】前記作業指示書に記述された利用順序は、コンピュータ名と、そのコンピュータ上で動作するアプリケーションプログラムおよび連携インタフェースからなるサービスエンティティの名称との組を該サービスエンティティの利用の順序に従って指定したものであり、前記コンピュータ名を省略可能とすることができる。

【００２５】さらに具体的には、本発明では、アプリケーションプログラム間連携処理を、アプリケーションプログラム間連携処理に関わる個々のアプリケーションプログラム本体にアプリケーションプログラム間連携処理手続きを実行するための連携インタフェースを付与したサービスエンティティ、連携させるサービスエンティティ間の移動先を表す利用順序とサービスエンティティの利用手続きを有する作業指示書（サービスエージェント）、このサービスエージェントをそれが有する利用順序に記述された移動先に移動させる通信路を生成、維持、管理する移動制御手段（ルーティングマネージャ）から構成する。

【００２６】サービスエンティティは、１つのアプリケーションプログラムの実行単位として分散処理システムを構成する個々のコンピュータ上に配置され、実行される。サービスエンティティを構成する連携インタフェースは、サービスエージェントの有する制御手続き、入出力データ手続き、表示手続きの中から現在のサービスエンティティに該当する手続きを選択し、その手続きを実行することにより、アプリケーションプログラム本体を実行制御する機能を持つ。また、実行する手続きに対応するプログラムが他のコンピュータ上にある場合には、そのプログラムをサービスエンティティが動作しているコンピュータ上にコピーし、そのプログラムを実行することにより、サービスエージェントの要求する手続きを実行する機能を持つ。

【００２７】アプリケーションプログラム間の連携処理を実現するサービスエージェントは、連携処理においてアプリケーションプログラムを連携させる順序を該当するアプリケーションプログラム本体を有するサービスエンティティの利用順序として記述した利用順序と、各サービスエンティティにおいて使用するアプリケーションプログラムの有する機能を利用する制御手続き、その制御手続きを用いてサービスエンティティの有するアプリケーションプログラムに実際に処理を行わせるために必要なデータを格納するデータ項目、データ項目に格納されたデータをアプリケーションプログラムに渡し、その処理結果データを該当するデータ項目に格納する入出力

9

データ処理手続き、サービスエージェントの有するデータをグラフィカルユーザインタフェースを通して表示するため表示手続きからなる利用手続きを有する。

【0028】ルーティングマネージャは、分散処理システムを構成する各コンピュータ上に1つずつ配置され、動作しており、各コンピュータ上で動作するルーティングマネージャ間でサービスエージェントが移動するための通信路を構築し、各コンピュータ上で動作するサービスエンティティの名称とそのサービスエンティティとの間でサービスエージェントを移動させるための通信路を管理する機能を持つ。さらに、他のルーティングマネージャから通信路を通して受け取ったサービスエージェントの必要としているサービスエンティティが自コンピュータ上で利用できるか否かを判定し、利用できる場合にはサービスエージェントをその必要としているサービスエンティティに送り、利用できない場合にはサービスエージェントを他のコンピュータ上のルーティングマネージャに移動させる機能を持つ。これらの機能を用いて、サービスエージェントの利用順序に従って、利用順序に指定されたサービスエンティティにサービスエージェントを移動させることにより、サービスエージェントが連携処理のために必要とするサービスエンティティ間の移動を行う。

【0029】

【作用】本発明では、アプリケーションプログラム間連携処理方法を上記の構成とすることにより、アプリケーションプログラム間の連携処理を作業指示書（サービスエージェント）というアプリケーションプログラム本体とは独立した形式で与えることができる。また、アプリケーションプログラム本体に連携処理のための連携インタフェースを付加し、この連携インタフェースがサービスエージェントの持つ利用手続きを実行することによりアプリケーションプログラム間の連携処理手続きをサービスエージェントを通して容易に変更することができる。

【0030】分散処理システムを構成する各コンピュータ上で動作する移動制御手段としてのルーティングマネージャは、サービスエージェントを送受信するための通信路を確立し、ルーティングマネージャの動作しているコンピュータ上で動作するサービスエンティティの名称とそのサービスエンティティとの間でサービスエージェントを送受信するための通信路を管理する。サービスエンティティが起動されると、サービスエンティティ（アプリケーションプログラム＋連携インタフェース）の動作するコンピュータ上のルーティングマネージャとの間にサービスエージェントを送受信するための通信路を確立し、ルーティングマネージャにサービスエンティティの名称を送り、これを受けるルーティングマネージャは送られたサービスエンティティの名称とそのサービスエンティティへの通信路を管理する。各ルーティングマネージャは、他のコンピュータ上で動作しているルーティングマネージャから通信路を通して受け取ったサービスエージェントの利用順序に指定されたサービスエンティティがそれ自身の動作するコンピュータ上で利用できるか否かを判定し、利用できる場合にはサービスエージェントを該当するサービスエンティティに送り、利用できない場合にはサービスエージェントを他のコンピュータ上のルーティングマネージャに移動させる。

10

【0031】サービスエンティティは、通信路を通してサービスエージェントを受け取ると、連携インタフェースがそのサービスエージェントの持つ利用手続きの中からそのサービスエンティティで使用される手続きとデータを取得し、取得した手続きを実行することにより、サービスエンティティの有するアプリケーションプログラム本体の処理を制御し、サービスエージェントの必要としている処理結果を得、その結果をサービスエージェントに送り、サービスエージェント中のデータ項目に格納する。サービスエージェントから取得した手続きに対応するプログラムが他のコンピュータ上にあることが指定されている場合には、連携インタフェースはそのプログラムをサービスエンティティが動作しているコンピュータ上にコピーし、そのプログラムを実行することにより、サービスエージェントの要求する手続きを実行する。サービスエージェントの必要としている処理結果を格納した後、サービスエンティティの連携インタフェースはサービスエージェントの移動先を利用順序にしたがって更新し、通信路を通じてルーティングマネージャにサービスエージェントを送る。ルーティングマネージャはサービスエンティティから送られたサービスエージェントの利用順序に指定されたサービスエンティティがそれ自身の動作するコンピュータ上で利用できるか否かを判定し、利用できる場合にはサービスエージェントを該当するサービスエンティティに送り、利用できない場合にはサービスエージェントを他のコンピュータ上のルーティングマネージャに移動させる。

【0032】このようにルーティングマネージャは、サービスエージェントの利用順序に従って分散処理システムを構成するコンピュータ間で利用順序に指定されたサービスエンティティが動作するコンピュータにサービスエージェントを移動させ、サービスエンティティは受け取ったサービスエージェントの持つ手続きを連携インタフェースが取得し、実行することにより、アプリケーションプログラム間の連携処理を行う。

【0033】本発明では、このようにアプリケーションプログラム間の連携処理手続きをサービスエージェントというアプリケーションプログラム本体とは独立した形式で与え、アプリケーションプログラム本体に連携処理のための連携インタフェースを付加し、この連携インタフェースがサービスエージェントの持つ利用手続きを実行することによりアプリケーションプログラム間の連携

処理手続きをサービスエージェントを通して容易に変更することができる。

【0034】この連携処理方法を用いることにより、業務に応じて、その業務を進めるために必要なさまざまなアプリケーションプログラムとその操作を組み合わせた利用手順を、サービスエージェントの中に一連の作業手順と各作業で使用されるアプリケーションプログラムの利用手続きとして記述し、このサービスエージェントをコンピュータネットワークに接続されたコンピュータ上で動作している各種サービスエンティティの間で移動させ、サービスエージェントを受信したサービスエンティティはサービスエージェントに記述された利用手続きを実行することにより、従来、作業者が行っていた複数のアプリケーションプログラムを組み合わせた利用や、各作業者に依頼して行っていたアプリケーションプログラムの操作、及びその結果の取得を自動化するとともに、一連の作業に必要なアプリケーションプログラムの利用手順をシステム化することにより、各作業者の作業負荷を軽減し、作業の流れを円滑にすることができる。

【0035】なお、本発明における好適なアプリケーションプログラムとしては、生産スケジュールの作成プログラム、生産システムの動作シミュレーションプログラムなどが考えられる。それぞれがコンピュータによって制御されている複数の製造設備から構成される工場の生産工程の管理を行なう際は、工場内の幾つかの生産物（ロット）に対し、次にどの設備で、どのような加工作業を行なわせるかを的確に指示する必要があり、従来はこのような指示を工程係が、時々刻々と変わる生産進行状況を判断して行なわなければならず、煩雑なうえ効率が悪いという問題があった。本発明によれば、上のような判断や指示をサービスを提供する設備とそれを受けるロットの間で情報を交換しながら自律的に（人手に依らず）行なうことが可能であり、以上のような生産システムの自律化などに好適である。また、本発明は、コンピュータネットワークを介して行う種々のサービス（オンラインショッピング、切符の予約など）への適用も可能である。

【0036】

【実施例】以下では、図面を使用して本発明の実施例について詳細に説明する。本実施例では、アプリケーションプログラム本体と連携インタフェースから構成される複数のサービスエンティティの間をサービスエージェントが移動しながら、サービスエンティティを通してアプリケーションプログラムを連携させ、処理を進めていくアプリケーションプログラム間連携処理方法について説明する。

【0037】図1は、本実施例のアプリケーションプログラム間連携処理方法の動作する分散処理システムの構成を表した図である。本実施例では、上記の分散処理システムにおいて、利用者が或る作業を行う際に、アプリケーションプログラムＡＰ１→ＡＰ２→ＡＰ３もしくはＡＰ４をこの順序で連携して処理させる必要があるものとする。また、各アプリケーションプログラムの処理においては、ＡＰ１はサービスエージェント（作業指示書：図３により後述）のデータ項目の項目１、項目２に格納されたデータを用いて処理を行い、その処理結果データを項目６に格納し、ＡＰ２は項目３、項目４に格納されたデータを用いて処理を行い、その処理結果データを項目７に格納し、ＡＰ３は項目４、項目６に格納されたデータを用いて処理を行い、その処理結果データを項目８に格納し、ＡＰ４は項目５に格納されたデータを用いて処理を行い、その処理結果データを項目９に格納するものとする。

【0038】本分散処理システムは、コンピュータ（1）、コンピュータ間でデータ通信を行うためのネットワーク（0）、各コンピュータ上で動作するアプリケーションプログラム本体（3）、他のアプリケーションプログラムと連携して処理を行うための連携インタフェース（３０）、アプリケーションプログラム本体（3）に連携インタフェース（３０）を付与したアプリケーションプログラム間連携処理の基本となる処理単位であるサービスエンティティ（4）、各アプリケーションプログラム間を移動し連携処理を実現するサービスエージェント（２０）、サービスエージェント（２０）の移動する通信路及びサービスエンティティとの間の通信路を確立、管理し、サービスエージェント（２０）をサービスエンティティ（4）間で移動させるルーティングマネージャ（１０）から構成される。

【0039】本実施例の説明上、コンピュータ（1）の名称をそれぞれＣ１、Ｃ２、Ｃ３とし、各アプリケーションプログラム本体（3）の名称をそれぞれＡＰ１、ＡＰ２、ＡＰ３、ＡＰ４とし、各アプリケーションプログラム本体ＡＰ１～ＡＰ４を有するサービスエンティティの名称をそれぞれＳＥ１、ＳＥ２、ＳＥ３、ＳＥ４とし、コンピュータＣ１～Ｃ３を区別する必要のある場合にはそれぞれ１ａ、１ｂ、１ｃの符号を、アプリケーションプログラム本体ＡＰ１～ＡＰ４を区別する必要のある場合にはそれぞれ３ａ、３ｂ、３ｃ、３ｄの符号を、サービスエンティティＳＥ１～ＳＥ４を区別する必要のある場合にはそれぞれ４ａ、４ｂ、４ｃ、４ｄの符号を用いる。

【0040】図２にコンピュータ（1）の構成例を示す。コンピュータ（1）は、キーボード（１１）やマウス（１２）などから構成される入力装置（１３）、ＣＰＵ（１４）やメモリ（１５）を格納したコンピュータ本体である処理装置（１６）、データを表示する表示装置（１７）、データやアプリケーションプログラムを格納する外部記憶装置（１８）、データを印刷するプリンタ（１９）から構成される。

【0041】入力装置（１３）は、上記以外のタブレッ

ト、タッチパネルでもよい。入力装置（１３）には、画像データを入力するためのスキャナ、音声を入力するためのマイクが加えられてもよい。入力装置（１３）のマウスは、表示装置（１７）上の位置を指定したり、表示装置（１７）に表示されたいくつかの選択肢を含むメニューの中から選択対象を指定するための手段であり、このような指定手段を有する他の装置、例えば光学式ペンやタッチパネルであってもよい。表示装置（１７）として、音声を出力するためのスピーカが加えられてもよい。

【００４２】ネットワーク（０）は、複数のコンピュータ（１）間でデータを送受信する手段であり、特定の場所内のネットワークである LAN（ Local Area Network ）、各拠点間のネットワークである WAN（ Wide Area Network ）、ネットワーク同士を相互に接続したネットワークであるインターネットのいずれのネットワークであってもよい。

【００４３】図１に示したルーティングマネージャ（１０）は、各コンピュータごとに１つずつ存在し、コンピュータ起動時に起動される。ルーティングマネージャ（１０）をコンピュータ起動時に起動する方法としては、例えば、ＵＮＩＸのデーモンを用いる方法などがある。

【００４４】本実施例によるアプリケーションプログラム間連携処理方法では、上記のネットワーク（０）に接続されたコンピュータ（１ａ）で作成されたサービスエージェント（２０）がルーティングマネージャ（１０）を通して、ネットワーク（０）に接続されたコンピュータ（１）間を移動し、これらのコンピュータ（１）上で動作するサービスエンティティ（４）の連携インタフェース（３０）を通してその中のアプリケーションプログラム本体（３）の処理を実行、制御し、アプリケーションプログラム間の連携処理を行う。このように、サービスエージェント（２０）を用い複数のサービスエンティティ（４）を通してアプリケーションプログラム本体（３）の実行の制御、管理を行い、一連の作業に用いられるアプリケーションプログラム間を連携して処理することにより、一連の作業を自動化、支援する。

【００４５】本実施例によるアプリケーションプログラム間連携処理方法では、一台以上のコンピュータ（１）上で動作する２つ以上のアプリケーションプログラム本体（３）を有するサービスエンティティ（４）で利用することが可能であるが、図１では３台のコンピュータと各々のコンピュータ上で動作する４つのサービスエンティティ（４）を、サービスエージェント（２０）を用いて連携させることを例にして説明する。このように、一台のコンピュータ（１）上には、複数のサービスエンティティ（４）を含むことができる。コンピュータ（１）やアプリケーションプログラム本体（３）を有するサービスエンティティ（４）の数は、図示の構成に限定されるものではない。

【００４６】本実施例によるアプリケーションプログラム間連携処理方法で用いられる典型的なサービスエージェント（２０）の構造を図３に示す。異なる複数個のサービスエージェント（２０）が同時にシステム内を流れうる。サービスエージェント（２０）は、個々の利用者がいずれかのコンピュータ上で作成する。また、これに加えて、作成されたサービスエージェント自体が、必要に応じて新たなサービスエージェントを生成することもある。

【００４７】図３に示すように、サービスエージェント（２０）は、個々のサービスエージェント（２０）を区別するための識別子（２０１）、サービスエージェントの移動先の一覧を記述した移動先リスト一覧（２０２）、サービスエージェントが移動してきた経路を管理する移動リスト一覧（２０３）、現時点でサービスエージェント（２０）が必要としているアプリケーションプログラム本体（３）を持つサービスエンティティ（４）の名称とそのサービスエンティティ（４）が動作しているコンピュータの名称を示す移動先名（２０４）、連携処理を行う複数のサービスエンティティ（４）が必要とする全てのデータ項目（２１１～２１９）を列挙した全データ項目（２１）、全データ項目（２１）の中の各データ項目（２１１～２１９）に応じた入力方法を記述したデータ入力手続き（２２）、全データ項目（２１）の中の各データ項目やサービスエンティティに応じた出力方法を記述したデータ出力手続き（２３）、データ入力手続き（２２）を用いて入力されたデータを処理する方法である入力データ処理手続き（２４）、このサービスエージェント（２０）の移動先を決定し、決定した移動先にサービスエージェント（２０）を送るための方法である移動手続き（２５）、各サービスエンティティ（４）の持つアプリケーションプログラム本体（３）の実行を制御するための手続きである制御手続き（２６）、新たなサービスエージェントを生成するための手続きである生成手続き（２７）から構成される。

【００４８】サービスエージェント（２０）の構造のうち、識別子（２０１）、移動先リスト一覧（２０２）、移動リスト一覧（２０３）、移動先名（２０４）は、サービスエージェント（２０）の作成時に必ず作成される項目である。上記以外の、全データ項目（２１）、データ入力手続き（２２）、データ出力手続き（２３）、入力データ処理手続き（２４）、移動手続き（２５）、制御手続き（２６）、生成手続き（２７）の各項目や手続きは、サービスエージェント（２０）を用いて行われるサービスエンティティ（４）間の連携処理の方法ごとに異なる要素であり、全ての要素が必ず存在するわけではない。

【００４９】全データ項目（２１）は、何等かの値を持つデータ項目（２１１～２１５）である入力済みデータ

15

項目（２７０）と、値を持たないデータ項目（２１６～２１９）である未入力データ項目（２８０）とから構成される。入力済みデータ項目（２７０）や未入力データ項目（２８０）は、サービスエージェント（２０）が或るサービスエンティティ（４）に受け取られた時に、そのサービスエンティティ（４）に渡すべきデータを格納しているデータ項目である出力データ項目（２７１）、或るサービスエンティティ（４）の実行結果データを格納するデータ項目である入力データ項目（２８１）を含んでいる。なお、ここでの入力および出力の別は、サービスエージェント（２０）から見たものであり、サービスエージェント（２０）からサービスエンティティ（４）へのデータの受け渡しを出力とし、その逆を入力としている。

【００５０】例えば、アプリケーションプログラム本体ＡＰ１が実行時に必要とするデータが項目１（２１１）、項目２（２１２）に格納されており、その実行結果データが項目６（２１６）に格納されるとすると、サービスエンティティＳＥ１では、項目１（２１１）と項目２（２１２）が出力データ項目（２７１）、項目６（２１６）が入力データ項目（２８１）である。アプリケーションプログラムＡＰ２が実行時に必要とするデータが項目３（２１３）、項目４（２１４）に格納されており、その実行結果データが項目７（２１７）に格納されるとすると、サービスエンティティＳＥ２では、項目３（２１３）と項目４（２１４）が出力データ項目、項目７（２１７）が入力データ項目である。同様に、アプリケーションプログラム本体ＡＰ３が実行時に必要とするデータが項目４（２１４）、項目６（２１６）に格納されており、その実行結果データが項目８（２１８）に格納されるとすると、サービスエンティティＳＥ３では、項目４（２１４）と項目６（２１６）が出力データ項目（２７１）、項目８（２１８）が入力データ項目（２８１）であり、アプリケーションプログラム本体ＡＰ４が実行時に必要とするデータが項目５（２１５）に格納されており、その実行結果データが項目９（２１９）に格納されるとすると、サービスエンティティＳＥ４では、項目５（２１５）が出力データ項目（２７１）、項目９（２１９）が入力データ項目（２８１）となる。このように出力データ項目（２７１）や入力データ項目（２８１）は、サービスエージェント（２０）が受け取られたサービスエンティティ（４）ごとに異なる。

【００５１】データ入力手続き（２２）には、未入力データ項目（２８０）の中からサービスエンティティ（４）の持つアプリケーションプログラム本体（３）に応じた入力データ項目（２８１）を選択するための手続きである入力データ選択手続き（２２１）、利用者により画面から入力された値を取得するための手続きである入力データ取得手続き（２２２）、アプリケーションプログラム本体（３）の出力結果データを取得するための手続きであるデータ取得手続き（２２３）、このデータ取得手続き（２２３）を用いて取得されたアプリケーションプログラム本体（３）の出力結果データを入力データ項目（２８１）のデータ形式に変換するための手続きである入力データ変換手続き（２２４）、入力データ取得手続き（２２２）を用いて得られた値や入力データ変換手続き（２２４）を用いて入力データ項目（２８１）のデータ形式に変換された値を入力データ項目（２８１）に格納するための手続きである入力データ格納手続き(２２５）がある。

16

【００５２】ここでいう「データ形式の変換」とは、データの書式の変換を意味する。具体的な例として、或るデータが属性名「個数」、「単価」を持ち、「個数」の値として「１０」、「３０」、及び「単価」の値として「２０」、「４０」を、アプリケーションプログラムが表１に示すような「アプリケーションプログラムの実行結果の出力書式」で出力し、入力データ項目には表２に示すような「サービスエージェントの入力データ項目の書式」で格納される場合、入力データ変換手続きは、以下のように、「アプリケーションプログラムの実行結果の出力書式」から「サービスエージェントの入力データ項目の書式」への書式の変換を行なう。

【００５３】

【表１】

・「アプリケーションプログラムの実行結果の出力書式」

```
((個数　単価)
  (10　20))
((個数　単価)
  (30　40)))
```

【００５４】

【表２】

・「サービスエージェントの入力データ項目の書式」

```
((個数　単価)
  (10　20)
  (30　40))
```

【００５５】データ出力手続き（２３）には、入力済みデータ項目（２７０）の中からサービスエンティティ（４）の持つアプリケーションプログラム本体（３）に応じた出力データ項目（２８１）を選択するための手続きである出力データ選択手続き（２３１）、この出力データ選択手続き（２３１）を用いて選択された出力データ項目（２８１）の値をアプリケーションプログラム本体（３）のデータ入力形式に変換するための手続きである出力データ変換手続き（２３２）、この出力データ変

換手続き（２３２）を用いて変換された出力データ項目（２８１）の値をアプリケーションプログラム本体（３）に渡すための手続きであるパラメタ設定手続き（２３３）、出力データ変換手続き（２３２）を用いて変換された出力データ項目（２８１）の値を画面に表示するための手続きである出力データ表示手続き（２３４）がある。

【００５６】入力データ処理手続き（２４）は、入力データ取得手続き（２２２）やデータ取得手続き（２２３）を用いて取得されたデータを処理するための手続きである。例えば、入力データ取得手続き（２２２）を用いて取得された複数のデータの和を計算する処理などがこの入力データ処理手続き（２４）として記述される。

【００５７】移動手続き（２５）は、サービスエージェント（２０）の移動先を決定し、決定した移動先にサービスエージェント（２０）を送るための手続きである。移動手続き（２５）の移動先の決定方法には、サービスエージェント作成時に与えられた移動順序に従って次の移動先を決定する方法、入力済みデータ項目（２７０）や未入力データ項目（２８０）の中の任意のデータ項目の組み合わせに応じてサービスエージェントの移動先を決定する方法がある。すなわち、サービスエージェント作成時に与えられた移動順序に従って次の移動先を決定する場合は、サービスエージェント（２０）の移動先リスト一覧（２０２）の先頭要素を取り出し、それを移動先名（２０４）に格納し、移動先リスト一覧（２０２）の値を更新し、更新したサービスエージェント（２０）を後述する入出力チャネル（４０３）を通してルーティングマネージャ（１０）に送る。任意のデータ項目の組合せに応じて移動先を決定する場合は、図２１に示すように、着目するデータ項目の値を調べ、その値により決定される移動先を移動先名（２０４）に格納し、移動先名を更新したサービスエージェント（２０）を入出力チャネル（４０３）を通してルーティングマネージャ（１０）に送る。

【００５８】制御手続き（２６）は、サービスエンティティ（４）の連携インタフェース（３０）で実行されるアプリケーションプログラム本体（３）の実行を制御するための手続きであり、アプリケーションプログラム本体（３）の起動・終了、アプリケーションプログラム本体（３）のコマンド実行、及び複数のコマンドの組み合わせである制御スクリプトがある。

【００５９】生成手続き（２７）は、新たなサービスエージェント（２０）を生成するための手続きであり、サービスエージェントを生成する制御スクリプトがある。

【００６０】図４～図６に、各コンピュータ上で動作するルーティングマネージャ（１０）により構築される論理的な通信路の接続形態を示す。ルーティングマネージャ（１０）は、複数のコンピュータ間の論理的通信路の確立と維持、各サービスエンティティ（４）との間の入出力チャネル（図６：４０３）の確立と維持を行う。論理的通信路は、相互に接続された２つのルーティングマネージャ（１０）を接続する論理的通信路であるチャネル（４０）の組み合わせから構成される。図４では、コンピュータＣ１、Ｃ２、Ｃ３上で動作する３つのルーティングマネージャ（１０）ＲＭ１、ＲＭ２、ＲＭ３の間で、ＲＭ１とＲＭ２、ＲＭ２とＲＭ３が接続され、ＲＭ１－ＲＭ２－ＲＭ３という論理的通信路を構成していることを表している。図５は、図４に示した論理的通信路を構成しているルーティングマネージャ（１０）ＲＭ２に新たなルーティングマネージャ（１０）ＲＭ４が接続され、論理的通信路の構成が変化した状態を示す図である。図６は、或る一台のコンピュータ内におけるルーティングマネージャ（１０）ＲＭ３とサービスエンティティ（４）ＳＥ３、ＳＥ４が入出力チャネル（４０３）により接続された接続状態、及びサービスエンティティ（４）を構成する連携インタフェース（３０）とアプリケーションプログラム本体（３）の間がアプリケーションインタフェース（３８）で接続された接続状態を表している。入出力チャネル（４０３）は、ルーティングマネージャ（１０）とサービスエンティティ（４）を接続した通信路であり、実際にはサービスエンティティ（４）内の連携インタフェース（３０）とルーティングマネージャ（１０）の間を接続する。アプリケーションインタフェース（３８）は、サービスエンティティ（４）内に生成される連携インタフェース（３０）とアプリケーションプログラム本体（３）の入出力を接続するインタフェースである。

【００６１】ルーティングマネージャ（１０）は、チャネル（４０）を接続するための受け口となるチャネルポート（４０１）を持つ。チャネルポートは個々のチャネルポート（４０１）を識別するためのチャネルポート番号を持つ。チャネルポート番号は、ルーティングマネージャ（１０）により、チャネルポート（４０１）生成時に与えられる。すなわち、チャネル（４０）とは２つのルーティングマネージャ（１０）のチャネルポート（４０１）を接続した通信路であり、チャネル（４０）を生成するとは、任意の２つのルーティングマネージャ（１０）のチャネルポート（４０１）の間を接続することである。

【００６２】ルーティングマネージャ（１０）は、チャネル（４０）を生成することによりチャネルポート（４０１）が使用されると新たなチャネルポート（４０２）を１つ生成する。そのためルーティングマネージャ（１０）は、常に他と接続されていない空きチャネルポート（４０２）を持つ。この空きチャネルポート（４０２）は、他のルーティングマネージャ（１０）から接続要求がなされた場合にそのルーティングマネージャ（１０）との間でチャネル（４０）を生成するために使用される。

19

【0063】ルーティングマネージャ（10）が生成した空きチャネルポート（402）は、サービスエンティティ（4）とルーティングマネージャ（10）を接続する際にも、ルーティングマネージャ（10）同士を接続する場合と同様に用いられる。すなわち、ルーティングマネージャ（10）とサービスエンティティ（4）との間に入出力チャネル（403）を生成する場合には、サービスエンティティ（4）が起動されたときに新たな空きチャネルポートをサービスエンティティ（4）の連携インタフェース（30）に生成し、サービスエンティティ（4）からルーティングマネージャ（10）に接続要求を行い、ルーティングマネージャ（10）が空きチャネルポート（402）とサービスエンティティ（4）の連携インタフェース（30）の空きチャネルポートの間に入出力チャネル（403）を作成し、ルーティングマネージャ（10）のみが新たな空きチャネルポート（402）を生成する。ルーティングマネージャ（10）は、このようにチャネル（40）が生成されると、新たな空きチャネルポート（402）を1つ生成し、常に1つの空きチャネルポート（402）が残るようにチャネルポートの管理をする。

【0064】サービスエンティティ（4）は、起動時にルーティングマネージャ（10）との間で入出力チャネル（403）を生成するための空きチャネルポート（402）を連携インタフェース（30）に生成し、ルーティングマネージャ（10）との間で入出力チャネル（403）を生成することに加え、連携インタフェース（30）とアプリケーションプログラム本体（3）の入出力を接続するためのアプリケーションインタフェース（38）を持ち、このアプリケーションインタフェース（38）を用いて連携インタフェース（30）とアプリケーションプログラム本体（3）の間の入出力、すなわち連携インタフェース（30）からアプリケーションプログラム本体（3）にコマンドを送ったり、アプリケーションプログラム本体（3）の実行結果データを連携インタフェース（30）に受け取ったりすること、を行う。

【0065】チャネルポート（401）は、受信時のバッファリング機能を持ち、チャネル（40）や入出力チャネル（403）を通して送られてきたデータを、読みだし要求があるまで保持する。

【0066】図7に、ルーティングマネージャ（10）を実現するための処理装置（16）におけるプログラムの構造を示す。ルーティングマネージャ（10）は、処理装置（16）のメモリ（15）上におかれる管理テーブル作成プログラム（41）、接続要求プログラム（42）、接続管理プログラム（43）、移動管理プログラム（44）、接続待ちプログラム（45）の各プログラムと、接続先管理テーブル（46）、サービスエンティティ管理テーブル（47）、接続管理テーブル（48）から構成される。

20

【0067】処理装置（16）のメモリ（15）上におかれたルーティングマネージャ（10）が起動されると、まず管理テーブル作成プログラム（41）が外部記憶装置（18）に格納されている接続先コンピュータの名称の一覧を読み込み、処理装置（16）のメモリ（15）上に接続先管理テーブル（46）を作成する。次にルーティングマネージャ（10）の動作するコンピュータで動作するサービスエンティティの名称の一覧を読み込み、処理装置（16）のメモリ（15）上にサービスエンティティ管理テーブル（47）を作成する。

【0068】接続要求プログラム（42）は、管理テーブル作成プログラム（41）により処理装置（16）のメモリ（15）上に作成された接続先管理テーブル（46）に格納されたコンピュータ名の一覧の先頭から順に接続先コンピュータ名を取得し、その接続先コンピュータに対して接続を要求する。接続ができない場合には次の接続先コンピュータ名を取得し、接続ができるまで順次接続要求を行う。接続先管理テーブル（46）に格納された接続先コンピュータ名がなくなるまで、順次、接続要求を行っても接続できない場合、エラーメッセージを出力し、処理を終了する。

【0069】接続管理プログラム（43）は、接続要求プログラム（42）により接続されたコンピュータ（接続先コンピュータ）と接続が完了した旨の報告を受け取ると、接続要求を行ったコンピュータ（接続元コンピュータ）のコンピュータ名を接続先コンピュータに送信し、接続先コンピュータから接続先コンピュータ名が送信されてくるのを待つ。接続先コンピュータから送られた接続先コンピュータ名を受け取ると、その接続先コンピュータの名称と接続先コンピュータに接続されたチャネル（40）のチャネルポート番号のペアを接続管理テーブル（48）に格納する。後述する接続待ちプログラム（45）がチャネルポート番号を受け取ったことにより制御が渡されてきた場合、そのチャネルポート番号から接続元コンピュータ名、もしくはサービスエンティティ（4）の名称が送られてくるのを待つ。接続元コンピュータ名、もしくはサービスエンティティ（4）の名称を受け取るとその名称とその間に接続された論理的通信路のチャネルポート番号のペアを接続管理テーブル（48）に格納し、送られてきた名称が接続元コンピュータ名の場合にはその接続元コンピュータに対して接続先コンピュータ名を送信する。

【0070】移動管理プログラム（44）は、接続管理プログラム（43）が接続元コンピュータ名またはサービスエンティティ（4）の名称とチャネルポート番号のペアを接続管理テーブル（48）に格納するか、或る一定時間の間に接続要求がなされなかった場合に起動される。移動管理プログラム（44）は、接続管理テーブル（48）の先頭から順にチャネルポート番号を取得し、そのチャネルポート番号（401）に接続されたチャネ

ル（40）からサービスエージェント（20）を読み出し、読み出したサービスエージェント（20）の移動先名（204）に指定されたサービスエンティティの名称が接続管理テーブル（48）に格納されているか否かをチェックする。格納されている場合には、そのサービスエージェント（20）をサービスエンティティ（4）に送る。格納されていない場合には、接続されている他のルーティングマネージャ（10）を接続管理テーブル（48）から検索し、その結果得られたルーティングマネージャ（10）にそのサービスエージェント（20）を送る。チャネル（40）から読み出すサービスエージェント（20）がない場合には、次のチャネルポート番号（401）に接続されたチャネル（40）から同様にサービスエージェント（20）を読み出す。これを接続管理テーブル（48）に格納されたチャネルポート番号（401）がなくなるまで繰り返す。接続管理テーブル（48）に格納されたチャネルポート番号の全てについて順次読み出しを行うと次の接続待ちプログラム（45）に制御を渡す。

【0071】接続待ちプログラム（45）は、或る一定時間他のルーティングマネージャ（10）もしくはサービスエンティティ（30）からの接続要求を待つ。或る一定時間の間に接続要求があれば、その接続要求元のルーティングマネージャ（10）との間でチャネル（40）を接続するか、接続要求元のサービスエンティティ（30）との間で入出力チャネル（403）を接続し、新たな空きチャネルポート（402）を生成し、新たに接続されたチャネル（40）の接続されたチャネルポート（401）のチャネルポート番号を接続管理プログラム（43）に送る。或る一定時間の間に接続要求がなければ、移動管理プログラム（44）に制御を渡す。

【0072】図8（a）に接続先管理テーブル（46）の構造を、図8（b）にサービスエンティティ管理テーブル（47）の構造を、図8（c）に接続管理テーブル（48）の構造を示す。

【0073】接続先管理テーブル（46）は、接続先のコンピュータの名称の一覧を保持する接続先コンピュータ名フィールド（460）を持つ。接続先コンピュータ名フィールド（460）は、ルーティングマネージャ（10）の起動時に接続要求を行うコンピュータの名称であるコンピュータ名（461）を値として持つ。この接続先管理テーブル（46）には、当然ながら自己のコンピュータの情報は含まない。

【0074】サービスエンティティ管理テーブル（47）は、ルーティングマネージャ（10）の動作するコンピュータ上で動作するサービスエンティティ（4）の名称の一覧を保持するサービスエンティティ名フィールド（470）を持つ。サービスエンティティ名フィールド（470）は、ルーティングマネージャ（10）の動作するコンピュータ上で動作するサービスエンティティ

（4）の名称（471）を値として持つ。

【0075】接続管理テーブル（48）は、チャネル（40）により接続された他のルーティングマネージャ（10）の動作するコンピュータの名称や入出力チャネル（403）により接続されたサービスエンティティ（4）の名称を保持する接続先名称フィールド（480）、それらの接続先との接続に用いられているチャネル（40）や入出力チャネル（403）が接続されているチャネルポート（401）のチャネルポート番号を保持するチャネルポート番号フィールド（481）、接続先がルーティングマネージャ（10）であるかサービスエンティティ（4）であるかを区別するための接続先種別フィールド（484）を持つ。接続先名称フィールド（480）は、ルーティングマネージャ（10）にチャネル（40）や入出力チャネル（403）を用いて接続されている接続先名（482）を値として持ち、チャネルポート番号フィールド（481）は、チャネル（40）や入出力チャネル（403）が接続されているチャネルポート番号（483）を接続先名（482）に対応付けて持つ。接続先種別フィールド（484）は、チャネルの接続先がルーティングマネージャであることを示すルーティングマネージャ、サービスエンティティであることを示すサービスエンティティのどちらか一方の値を取る。この2つの値をあわせて接続先種別名（485）と呼ぶ。

【0076】図9に連携インタフェース（30）を実現するための処理装置（16）におけるプログラムの構造を示す。連携インタフェース（30）は、処理装置（16）のメモリ（15）上におかれるデータ作成プログラム（31）、データ出力プログラム（32）、実行制御プログラム（33）、データ取得プログラム（34）、データ格納プログラム（35）、移動先操作プログラム（36）、関数制御プログラム（37）の各プログラムと、関数テーブル（371）とから構成される。関数テーブル（371）を設けることにより、サービスエージェント（20）自体にプログラムを保持する必要がなくなり、コンピュータ間のサービスエージェント（20）の伝送時の負荷を軽減することができる。

【0077】連携インタフェース（30）は、入出力チャネル（403）を通じてサービスエージェント（20）を受け取ると、受け取ったサービスエージェント（20）をデータ作成プログラム（31）に渡す。

【0078】データ作成プログラム（31）は、受け取ったサービスエージェント（20）の中の出力データ選択手続き（231）を用いて、入力済みデータ項目（270）の中からアプリケーションプログラム本体（3）に渡すデータを格納しているデータ項目である出力データ項目（271）を選択する。

【0079】出力データ項目（271）が決定されると、サービスエージェント（20）の中の出力データ変

換手続き（２３２）を用いて、出力データ項目（２７１）に格納されているデータからアプリケーションプログラム本体（３）に渡すデータ、もしくは画面に出力するデータを作成し、どちらのデータであるかのタグを付加し、データ出力プログラム（３２）に作成したデータを渡す。

【００８０】データ出力プログラム（３２）にデータが渡されると、データ出力プログラム（３２）は受け取ったデータのタグからデータを画面に出力するか、アプリケーションプログラム本体（３）に送るかを判定し、画面に出力する場合にはサービスエージェント（２０）の出力データ表示手続き（２３４）を用いて画面に出力する。アプリケーションプログラム本体（３）に渡す場合、データを実行制御プログラム（３３）に渡す。サービスエージェント（２０）に記述された出力データ表示手続き（２３４）が、プログラムではなく単なる関数名だけの場合、データ出力プログラム（３２）は、関数制御プログラム（３７）にその関数名を渡す。

【００８１】関数制御プログラム（３７）は、関数名を受け取ると、関数テーブル（３７１）からその関数名に該当する関数を検索し、検索したプログラムを実行する。実行制御プログラム（３３）は、データ出力プログラム（３２）からのデータを受け取ると、サービスエージェント（２０）の制御手続き（２６）を実行してアプリケーションプログラム本体（３）の実行を制御する。アプリケーションプログラム本体（３）を実行する際に、アプリケーションプログラム本体（３）にデータを渡す必要がある場合には、サービスエージェント（２０）のパラメタ設定手続き（２３３）を用いてアプリケーションプログラム本体（３）にデータを送る。アプリケーションプログラム本体（３）の実行を制御し、制御手続きの実行が終了すると、サービスエージェント（２０）のデータ取得手続き（２２３）を用いてアプリケーションプログラム本体（３）の実行結果データを取得し、取得したデータをデータ取得プログラム（３４）に渡す。制御手続き（２６）が関数名のみであったり、制御手続き（２６）の中に関数名のみの処理が含まれている場合には、その関数名を関数制御プログラム（３７）に渡し、該当する関数のプログラムを実行する。

【００８２】データ取得プログラム（３４）は、実行結果データを受け取るとサービスエージェント（２０）の入力データ選択手続き（２２１）を用いてサービスエージェント（２０）の全データ項目（２１）の中からアプリケーションプログラム本体（３）の実行結果データを格納するデータ項目である入力データ項目（２８１）を選択する。画面が出力されている場合には、サービスエージェント（２０）の入力データ取得手続き（２２２）を用いて画面に入力されたデータのうち、必要なデータのみを取得する。これらの取得したデータ（アプリケーションプログラム本体（３）の実行結果データ、画面からの入力データ）と入力データ項目（２８１）を、データ格納プログラム（３５）に渡す。サービスエージェント（２０）に記述された入力データ取得手続き（２２２）が、プログラムではなく単なる関数名だけの場合、データ取得プログラム（３４）は、関数制御プログラム（３７）にその関数名を渡し、該当する関数名の関数のプログラムを実行する。

【００８３】データ格納プログラム（３５）は、データ取得プログラム（３４）からのデータと入力データ項目（２８１）を受け取ると、サービスエージェント（２０）の入力データ処理手続き（２４）の領域を参照し、入力データ処理手続き（２４）があれば、その入力データ処理手続き（２４）を用いて、受け取ったデータ及び全項目データ（２１）に格納されているデータを処理し、処理後のデータを入力データ変換手続き（２２４）を用いて、受け取った入力データ項目（２８１）のデータ形式に変換し、変換したデータを入力データ項目（２８１）のそれぞれのデータ項目に格納する。入力データ処理手続き（２４）がない場合には、入力データ変換手続き（２２４）を用いて、受け取った入力データ項目（２８１）のデータ形式に変換し、変換したデータを入力データ項目（２８１）のそれぞれのデータ項目に格納する。

【００８４】移動先操作プログラム（３６）は、入力データ項目（２８１）にデータが格納されると呼び出され、移動先リスト一覧（２０２）の先頭要素を取りだしそれを移動先名（２０４）に格納し、このようにして移動先名（２０４）を更新したサービスエージェント（２０）を入出力チャネル（４０３）を通してルーティングマネージャ（１０）に送る。

【００８５】図１０（ａ）に関数制御プログラム（３７）の使用する関数テーブル（３７１）の構造を示す。関数テーブル（３７１）は、関数名フィールド（３７２）と関数の処理プログラム本体である処理フィールド（３７３）からなる。処理フィールド（３７３）は、関数の処理内容を記述したプログラム（３７３ａ）、または、関数の処理内容を記述したプログラムの格納位置（３７３ｂ）のいずれかが記述される。

【００８６】関数制御プログラム（３７）は、該当する関数名の処理フィールド（３７３）がプログラム（３７３ａ）であれば、そのプログラムを実行する。プログラムの格納位置（３７３ｂ）である場合、その格納位置にあるプログラム（３７４）をネットワーク（０）を通じて処理装置（１６）のメモリ（１５）上に読み込んだ（コピーした）後、そのプログラムを実行する。このような関数テーブルにプログラム自体ではなくプログラムの格納位置を保持しておくことにより、他のコンピュータ上にあるプログラムを共用することができる。これは、関数テーブルのメモリ容量を軽減するとともに、プログラムの変更時に１カ所のプログラムのみの変更で済

むという利点を有する。

【0087】プログラムの格納位置（373b）は、図10（b）の記述形式により指定される。格納位置は、プログラムの格納されているコンピュータの名称（375a）とコンピュータ名（375a）で指定されたコンピュータ上のプログラムの格納位置を表す格納パス（375c）、コンピュータ名（375a）と格納パス（375c）を区別するための区切り記号（375b）を用いて、コンピュータ名（375a）、区切り記号：（375b）、格納パス（375c）の形式で指定する。

【0088】図11（a）にサービスエージェント（20）の移動先リスト一覧（202）に格納される移動先リスト（91）の記述形式を、その具体的記述例を図11（b）に示す。

【0089】移動先リスト（91）は、サービスエージェント（20）を移動させる先のコンピュータの名称である移動先コンピュータ名（912）、移動先のコンピュータで利用するサービスエンティティ（4）の名称であるサービスエンティティ名（913）の組みである移動先（911）を、サービスエージェント（20）の移動順序に従って矢印（→）でつないだものである。移動先のコンピュータ名が分からない場合には、移動先コンピュータ名（912）にUnknownを指定する。

【0090】図11（b）の例では、サービスエージェント（20）は、C1という名称のコンピュータのSE1という名称のサービスエンティティ（4）の持つアプリケーションプログラム本体（3）AP1を用いて処理を行った後、C2という名称のコンピュータのSE2という名称のサービスエンティティ（4）の持つアプリケーションプログラム本体（3）AP2を用いて処理を行い、次にC3という名称のコンピュータのSE3という名称のサービスエンティティ（4）の持つアプリケーションプログラム本体（3）AP3を用いて処理を行い、最後にサービスエンティティ（4）SE4の動作するコンピュータ名は不明であるが、SE4という名称のサービスエンティティ（4）の持つアプリケーションプログラム本体（3）AP4を用いて処理を行うことを示している。

【0091】図12（a）にサービスエージェント（20）の移動リスト一覧（203）に格納される移動リストの記述形式（101）を、その具体的記述例を図12（b）に示す。

【0092】移動リスト一覧（203）は、サービスエージェント（20）が利用するサービスがネットワーク上のどこのコンピュータ上で動作しているかを探して移動する際に用いる。移動リスト一覧（203）は、具体的には、図11で示した移動先リストの移動先コンピュータ名（912）がUnknownの時に用いられる。移動先コンピュータ名（912）がUnknownの場合、各コンピュータのルーティングマネージャ（10）

は、サービスエージェント（20）の移動先名（204）に指定されている名称を持つサービスエンティティ（4）が存在するか否かをチェックし、存在しない場合、そのコンピュータの名称を移動コンピュータ名（102）に格納し、他のコンピュータ上のルーティングマネージャ（10）にサービスエージェント（20）を送る。

【0093】移動リスト（101）は、移動先コンピュータ名（912）がUnknownの場合に、サービスエージェント（20）の移動先名（204）に指定されているサービスエンティティ（4）が存在しないコンピュータの名称をサービスエージェント（20）が移動してきた順序に従って矢印（→）でつないだものである。移動リスト（101）は、サービスエージェント（20）が、移動するたびにそのコンピュータ名が順次、追加されていき、サービスエージェント（20）の移動先名（204）に指定されているサービスエンティティ（4）が動作するコンピュータが見つかった場合にクリアされる。このように、一度探索したコンピュータをこのリスト101に保存し、同一のコンピュータを何度も検索するのを防ぐことができる。移動リスト一覧（203）は、これが無い場合、エージェント（20）が利用するサービスを探して同じコンピュータを何度も訪れる可能性があり、このような事態を回避して効率的なサービスの探索を行なう為には有用なものである。

【0094】図12（b）の例では、サービスエージェント（20）の移動先名（204）に指定されているサービスエンティティ（4）がC1という名称のコンピュータに存在せず、次にC2という名称のコンピュータにも存在しなかったことを示している。

【0095】サービスエージェント（20）を用いたアプリケーションプログラム間の連携処理方法を実現するためにルーティングマネージャ（10）が行うサービスエージェント（20）の移動処理手続きのフローを図13～図16に示す。

【0096】サービスエージェント（20）は、それが作成されたコンピュータのルーティングマネージャ（10）に送られる。この時点で、サービスエージェント（20）の移動先リスト一覧（202）の値は図11（b）に示した移動順序であるとする。この時のサービスエージェント（20）の移動先名（204）の値は、移動先リスト一覧（202）の値である移動先リスト（91）の先頭の要素SE1（911a）となる。

【0097】ルーティングマネージャ（10）は、サービスエージェント（20）を受け取る（1100）と、まずサービスエージェント（20）の移動先名（204）の移動先コンピュータ名がルーティングマネージャ（10）の動作しているのコンピュータ名と同じか否かをチェックする（1101）。

【0098】サービスエージェント（20）の移動先名

27

（２０４）の移動先コンピュータ名とルーティングマネージャ（１０）の動作しているコンピュータ名が異なる場合、接続管理テーブル（４８）の接続先種別フィールド（４８４）の接続先種別名（４８５）がルーティングマネージャである接続先の中から、接続先名称フィールド（４８０）の接続先名（４８２）が移動先コンピュータ名と同じ接続先が登録されているか否かをチェックする（１１０２）。移動先コンピュータ名が登録されている場合、その移動先コンピュータ名に該当する接続先名（４８２）のチャネルポート番号フィールド（４８１）に格納されているチャネルポート番号（４８３）を取りだし（１１０３）、そのチャネルポート番号（４８３）に接続されているチャネル（４０）を通して移動先コンピュータのルーティングマネージャ（１０）にサービスエージェント（２０）を送る（１１０４）。登録されていない場合、接続管理テーブル（４８）の接続先種別フィールド（４８４）の接続先種別名（４８５）がルーティングマネージャである接続先の中から、最初に登録されている接続先名（４８２）のチャネルポート番号フィールド（４８１）に格納されているチャネルポート番号（４８３）を取りだし（１１０５）、そのチャネルポート番号に接続されたチャネル（４０）を通してチャネルに接続されている登録先コンピュータのルーティングマネージャ（１０）にサービスエージェント（２０）を送る（１１０６）。

【００９９】サービスエージェント（２０）の移動先名（２０４）の移動先コンピュータ名（９１２）とルーティングマネージャ（１０）の動作しているコンピュータ名が同じである場合、接続管理テーブル（４８）の接続先種別フィールド（４８４）に格納されている接続先種別名（４８５）がサービスエンティティである接続先の中から接続先名称フィールド（４８０）の接続先名（４８２）が移動先名（２０４）のサービスエンティティ名（９１３）と同じサービスエンティティが登録されているか否かをチェックする（１１０７）。登録されていれば、その登録されているサービスエンティティのチャネルポート番号フィールド（４８１）に格納されているチャネルポート番号（４８３）を取得し（１１０８）、そのチャネルポート番号（４８３）に接続されている入出力チャネル（４０３）を通して、サービスエンティティ（４）の連携インタフェース（３０）にサービスエージェント（２０）を送る（１１０９）。

【０１００】サービスエンティティ名が接続管理テーブル（４８）に登録されていなければ、サービスエンティティ管理テーブル（４７）にサービスエンティティ名が登録されているか否かをチェックする（１１１０）。サービスエンティティ管理テーブル（４７）にサービスエンティティ名が登録されていれば、そのサービスエンティティ（４）を起動する（１１１１）。起動されたサービスエンティティ（４）の連携インタフェース（３０）

28

は、ルーティングマネージャ（１０）に接続要求を行う。ルーティングマネージャ（１０）は、サービスエンティティ（４）の連携インタフェース（３０）からの接続要求を待ち（１１１２）、接続要求を受け取るとルーティングマネージャ（１０）と連携インタフェース（３０）間の入出力チャネル（４０３）を生成し（１１１３）、起動したサービスエンティティ名とその入出力チャネル（４０３）のチャネルポート番号を接続管理テーブル（４８）に登録し（１１１４）、生成した入出力チャネル（４０３）を通してサービスエージェント（２０）をサービスエンティティ（４）の連携インタフェース（３０）に送る（１１０９）。サービスエンティティ管理テーブル（４７）にサービスエンティティ名が登録されていなければ、サービスエージェント（２０）の移動先名（２０４）の移動先コンピュータ名（９１２）をＵｎｋｎｏｗｎに変更し（１１１５ａ）、接続管理テーブル（４８）の接続先種別フィールド（４８４）の接続先種別名（４８５）がルーティングマネージャである接続先の中から、最初に登録されている接続先名（４８２）のチャネルポート番号フィールド（４８１）に格納されているチャネルポート番号（４８３）を取りだし（１１０５）、そのチャネルポート番号に接続されたチャネル（４０）を通してチャネルに接続されている登録先コンピュータのルーティングマネージャ（１０）にサービスエージェント（２０）を送る（１１０６）。

【０１０１】サービスエージェント（２０）の移動先名（２０４）の移動先コンピュータ名（９１２）がＵｎｋｎｏｗｎの場合、図１４に移り、移動先名（２０４）のサービスエンティティ名が、接続管理テーブル（４８）の接続先種別フィールド（４８４）の接続先種別名（４８５）がサービスエンティティである接続先の中から接続先名称フィールド（４８０）の接続先名（４８２）がサービスエージェント（２０）の移動先名（２０４）のサービスエンティティ名と同じ接続先が登録されているか否かをチェックする（１１１５ｂ）。登録されていれば、その登録されているサービスエンティティのチャネルポート番号フィールド（４８１）に格納されているチャネルポート番号（４８３）を取得し（１１１６）、取得したチャネルポート番号（４８３）に接続されている入出力チャネル（４０３）を通して、サービスエンティティ（４）の連携インタフェース（３０）にサービスエージェント（２０）を送る（１１１７）。サービスエンティティ名が接続管理テーブル（４８）に登録されていなければ、次にサービスエンティティ管理テーブル（４７）にサービスエンティティ名が登録されているか否かをチェックする（１１１８）。サービスエンティティ管理テーブル（４７）にサービスエンティティ名が登録されていれば、そのサービスエンティティ（４）を起動する（１１１９）。起動されたサービスエンティティ（４）の連携インタフェース（３０）は、ルーティング

29

マネージャ（１０）に接続要求を行う。ルーティングマネージャ（１０）は、サービスエンティティ（４）の連携マネージャ（３０）からの接続要求を待ち（１１２０）、接続要求を受け取るとルーティングマネージャ（１０）と連携インタフェース（３０）間の入出力チャネル（４０３）を生成し（１１２１）、起動したサービスエンティティ名とその入出力チャネル（４０３）のチャネルポート番号を接続管理テーブル（４８）に登録し（１１２２）、生成した入出力チャネル（４０３）を通してサービスエージェント（２０）をサービスエンティティ（４）の連携インタフェース（３０）に送る（１１１７）。

【０１０２】接続管理テーブル（４８）、サービスエンティティ管理テーブル（４７）のいずれのテーブルにも登録されていなければ、図１５に移り、接続管理テーブル（４８）の接続先種別フィールド（４８４）の接続先種別名（４８５）がルーティングマネージャである接続先があるか否かチェックする（１１２３）。接続先が登録されていれば、その中の最初に登録されている接続先名（４８２）を取りだし（１１２４）、その接続先名（４８２）が移動リスト一覧（２０３）に格納された移動リスト（１０１）の移動コンピュータ名（１０２）にあるか否かをチェックする（１１２５）。移動リスト（１０１）になければ、移動リスト（１０１）の移動コンピュータ名（１０２）に現在のコンピュータの名称を追加し（１１２６）、追加した移動リスト（１０１）をサービスエージェント（２０）の移動リスト一覧（２０３）に格納し（１１２７）、その接続先名（４８２）のチャネルポート番号フィールド（４８１）に格納されているチャネルポート番号（４８３）を取りだし（１１２８）、そのチャネルポート番号に接続されたチャネル（４０）を通してチャネルに接続されているコンピュータのルーティングマネージャ（１０）にサービスエージェント（２０）を送る（１１２９）。移動リスト（１０１）にあれば、接続管理テーブル（４８）の接続先種別フィールド（４８４）の接続先種別名（４８５）がルーティングマネージャである次の接続先があるか否かチェックする（１１２３）。次の接続先が接続管理テーブル（４８）に登録されていれば、その接続先名（４８２）を取りだし（１１２４）、その接続先名（４８２）がサービスエージェント（２０）の移動リスト一覧（２０３）に格納されている移動リスト（１０１）の移動コンピュータ名（１０２）にあるか否かをチェックする（１１２５）。これを接続管理テーブル（４８）の接続先種別フィールド（４８４）の接続先種別名（４８５）がルーティングマネージャである接続先がなくなるか、移動リスト（１０１）にない接続先名（４８２）が見つかるまで順次繰り返す。その結果、接続管理テーブル（４８）の接続先種別フィールド（４８４）の接続先種別名（４８５）がルーティングマネージャである接続先がな

30

くなった場合、エラーとしてそのサービスエージェント（２０）の移動を終了する（１１３０）。

【０１０３】図１６に移り、サービスエージェント（２０）を受け取ったサービスエンティティ（４）の連携インタフェース（３０）は、受け取ったサービスエージェント（２０）に記述されているデータ出力手続き（２３）を実行しサービスエージェント（２０）からサービスエンティティ（４）に送るデータを取得し（１１３１）、制御手続き（２６）を実行して、そのサービスエンティティ（４）の持つアプリケーションプログラム本体（３）の実行を制御し（１１３２）、その実行結果データをサービスエージェント（２０）のデータ入力手続き（２２）を実行してサービスエージェント（２０）に格納し（１１３３）、サービスエージェント（２０）の移動手続き（２５）を実行して移動先名（２０４）の値を次の移動先に書き換え（１１３４）、移動先名（２０４）の値を書き換えたサービスエージェント（２０）を、連携インタフェース（３０）からルーティングマネージャ（１０）へ入出力チャネル（４０３）を通して送る（１１３５）。図１３に戻り、ルーティングマネージャ（１０）は、連携インタフェース（３０）から送られたサービスエージェント（２０）を入出力チャネル（４０３）を通して受け取る（１１００）と、チャネル（４０）を通してサービスエンティティ（２０）を受け取ったときと同様に、移動先名（２０４）の移動先コンピュータ名（９１２）をチェックし（１１０１）、サービスエージェント（２０）を次の移動先にチャネル（４０）を通して送る（１１０４）。このサービスエージェントの移動処理をサービスエージェント（２０）の移動先リスト（９１）の最後の要素になるまで、ルーティングマネージャ（１０）間で繰り返すことにより、サービスエージェント（２０）をサービスエンティティ（４）間で移動させ、移動させたサービスエージェント（２０）をサービスエンティティ（４）の連携インタフェース（３０）で処理することにより各アプリケーションプログラム本体（３）の実行を制御する。

【０１０４】図１７～図２０にサービスエンティティ（４）の連携インタフェース（３０）において、受け取ったサービスエージェント（２０）の持つデータや手続きを利用しながら、アプリケーションプログラム本体（３）の実行を制御するための処理フローを示す。

【０１０５】まず、図１７において、連携インタフェース（３０）は、入出力チャネル（４０３）を通じてルーティングマネージャ（１０）からサービスエージェント（２０）を受け取る（１２００）と、受け取ったサービスエージェント（２０）の出力データ選択手続き（２３１）を実行して入力済みデータ項目（２７０）の中からアプリケーションプログラム本体（３）に渡すデータを格納しているデータ項目である出力データ項目（２７１）を決定する（１２０１）。

【0106】出力データ項目（271）が決定されると、サービスエージェント（20）の出力データ変換手続き（232）がプログラムであるか関数名であるかをチェックする（1202）。出力データ変換手続き（232）がプログラムであれば、その出力データ変換手続き（232）がアプリケーションプログラム本体（3）へのデータ出力関数か画面出力関数かをチェックし（1203）、データ出力関数であれば、その関数を実行して、出力データ項目（271）に格納されているデータからアプリケーションプログラム本体（3）に渡すデータを生成する（1204）。画面出力関数であれば、その関数を実行して、出力データ項目（271）に格納されているデータから画面に出力するデータを作成する（1205）。生成されたデータには、どちらのデータであるかを示すタグが付加される（1206）。出力データ変換手続き（232）が関数名であれば、その関数名のプログラムを関数制御プログラム（37）を通して関数テーブルから取得し（1207）、取得した関数がアプリケーションプログラム本体（3）へのデータ出力関数か画面出力関数かをチェックする（1203）。

【0107】タグが付加されたデータは、タグからデータを画面に出力するか、アプリケーションプログラム本体（3）に渡すかが判定され（1208）、画面に出力するデータの場合にはサービスエージェント（20）の出力データ手続き（234）がプログラムであるか関数名であるかがチェックされる（1209）。出力データ手続き（234）がプログラムであれば、その出力データ手続き（234）を実行し、画面出力データを画面に出力する（1210）。出力データ手続き（234）が関数名であれば、その関数名のプログラムを関数制御プログラム（37）を通して関数テーブルから取得し（1211）、取得した関数を実行し、画面出力データを画面に出力する（1210）。アプリケーションプログラム本体（3）に渡すデータの場合、そのデータのタグを削除する（1212）。

【0108】次に図18に移り、サービスエージェント（20）の制御手続き（26）がプログラムであるか関数名であるかがチェックされる（1213）。制御手続き（26）がプログラムであれば、その制御手続き（26）を実行することにより、アプリケーションプログラム本体（3）の実行を制御する（1214）。制御手続き（26）が関数名であれば、その関数名のプログラムを関数制御プログラム（37）を通して関数テーブルから取得し（1215）、取得した関数を実行することにより、アプリケーションプログラム本体（3）の実行を制御する（1214）。

【0109】制御手続き（26）によりアプリケーションプログラム本体（3）の実行制御が終了すると、サービスエージェント（20）のデータ取得手続き（223）がプログラムであるか関数名であるかがチェックされる（1216）。データ取得手続き（223）がプログラムであれば、そのデータ取得手続き（223）を実行し、アプリケーションプログラム本体（3）の実行結果データを取得する（1217）。データ取得手続き（223）が関数名であれば、その関数名のプログラムを関数制御プログラム（37）を通して関数テーブルから取得し（1218）、取得した関数を実行し、アプリケーションプログラム本体（3）の実行結果データを取得する（1217）。

【0110】画面が出力されている場合には、図20に移り、サービスエージェント（20）の制御手続き（26）がプログラムであるか関数名であるかがチェックされる（1219）。制御手続き（26）がプログラムであれば、その制御手続き（26）を実行することにより、画面からの入出力を制御する（1220）。制御手続き（26）が関数名であれば、その関数名のプログラムを関数制御プログラム（37）を通して関数テーブルから取得し（1221）、取得した関数を実行することにより、画面からの入出力を制御する（1220）。

【0111】画面が出力されている場合には、データ取得手続き（223）がプログラムであるか関数名であるかがチェックされる（1222）。データ取得手続き（223）がプログラムであれば、そのデータ取得手続き（223）を実行し、画面からの入力データを取得する（1223）。すなわち、画面を介してユーザの入力する指示やデータを取り込む。データ取得手続き（223）が関数名であれば、その関数名のプログラムを関数制御プログラム（37）を通して関数テーブルから取得し（1224）、取得した関数を実行し画面からの入力データを取得する（1223）。

【0112】次に図18に戻り、サービスエージェント（20）の入力データ選択手続き（221）がプログラムであるか関数名であるかがチェックされる（1225）。入力データ選択手続き（221）がプログラムであれば、その入力データ選択手続き（221）を実行し、サービスエージェント（20）の全データ項目（21）の中からアプリケーションプログラム本体（3）の処理結果データを格納するデータ項目である入力データ項目（281）を選択する（1226）。入力データ選択手続き（221）が関数名であれば、その関数名のプログラムを関数制御プログラム（37）を通して関数テーブルから取得し（1227）、取得した関数を実行し、サービスエージェント（20）の全データ項目（21）の中からアプリケーションプログラム本体（3）の処理結果データを格納するデータ項目である入力データ項目（281）を選択する（1226）。

【0113】次に図19に移り、サービスエージェント（20）の入力データ処理手続き（24）がプログラムであるか関数名であるかがチェックされる（1228）。入力データ処理手続き（24）がプログラムであ

れば、その入力データ処理手続き（２４）を実行し、受け取ったデータ及び全項目データ（２１）に格納されているデータを処理する（１２２９）。入力データ処理手続き（２４）が関数名であれば、その関数名のプログラムを関数制御プログラム（３７）を通して関数テーブルから取得し（１２３０）、取得した関数を実行し、受け取ったデータ及び全項目データ（２１）に格納されているデータを処理する（１２２９）。

【０１１４】次にサービスエージェント（２０）の入力データ変換手続き（２２４）がプログラムであるか関数名であるかがチェックされる（１２３１）。入力データ変換手続き（２２４）がプログラムであれば、その入力データ変換手続き（２２４）を実行し、受け取ったデータをそれに該当する入力データ項目（２８１）のデータ形式に変換し（１２３２）、変換したデータを入力データ項目（２８１）のそれぞれのデータ項目に格納する（１２３４）。入力データ変換手続き（２２４）が関数名であれば、その関数名のプログラムを関数制御プログラム（３７）を通して関数テーブルから取得し（１２３３）、取得した関数を実行し受け取ったデータをそれに該当する入力データ項目（２８１）のデータ形式に変換し（１２３２）、変換したデータを入力データ項目（２８１）のそれぞれのデータ項目に格納する（１２３４）。

【０１１５】入力データ項目（２８１）にデータが格納されると、サービスエージェント（２０）の移動手続き（２５）がプログラムであるか関数名であるかがチェックされる（１２３５）。移動手続き（２５）がプログラムであれば、その移動手続き（２５）を実行し、移動先名（２０４）の値を次の移動先に書き換え（１２３６）、移動先名（２０４）の値を書き換えたサービスエージェント（２０）を、連携インタフェース（３０）からルーティングマネージャ（１０）へ入出力チャネル（４０３）を通して送る（１２３７）。移動手続き（２５）が関数名であれば、その関数名のプログラムを関数制御プログラム（３７）を通して関数テーブルから取得し（１２３８）、取得した関数を実行しを実行し、移動先名（２０４）の値を次の移動先に書き換え（１２３６）、移動先名（２０４）の値を書き換えたサービスエージェント（２０）を、連携インタフェース（３０）からルーティングマネージャ（１０）へ入出力チャネル（４０３）を通して送る（１２３７）。

【０１１６】次に、本発明の具体的な適用例として、製造ライン管理システムへ適用した例について説明する。製造ライン管理システムでは、製造・検査機器、それら機器の制御用アプリケーションプログラム、生産管理に必要なアプリケーションプログラムを製造物ごとに決められた順序で連携させる必要がある。

【０１１７】図２２に、本発明による製造ライン管理システムの一適用例のシステム構成を示す。本適用例では、製造ライン管理システムが各製造工程Ａ、Ｂを管理するコンピュータＣ２（１３４２）、Ｃ３（１３４３）、各製造工程Ａ、Ｂに接続された製造機器ＭＡ１（１３３２）、ＭＢ１（１３３３）と、これらの製造機器ＭＡ１（１３３２）、ＭＢ１（１３３３）を制御するアプリケーションプログラムＡＰ２（１３０２）、ＡＰ３（１３０３）、コンピュータＣ１（１３４１）上で製造指示書を作成するアプリケーションプログラムＡＰ１（１３０１）、コンピュータを接続するネットワーク（０）から構成されるものとし、コンピュータＣ１（１３４１）上のアプリケーションプログラムＡＰ１（１３０１）で作成された製造指示書は、各製造工程に接続された製造機器をそれらの制御アプリケーションプログラムを通して連携制御することにより、製造を進めていく場合を例に取り説明する。

【０１１８】本実施例による製造ライン管理システムでは、これらの各種アプリケーションプログラムＡＰ１（１３０１）、ＡＰ２（１３０２）、ＡＰ３（１３０３）をサービスエンティティＳＥ１（１３２１）、ＳＥ２（１３２２）、ＳＥ３（１３２３）（各サービスエンティティの名称はそれぞれＳＥ１、ＳＥ２、ＳＥ３とする）として実現する。製造物に添付される製造指示書はサービスエージェントＳＡ１（１４００）として具現する。製造指示書を具現したサービスエージェントＳＡ１（１４００）は、サービスエンティティＳＥ１（１３２１）を用いて利用者により作成され、まず製造工程Ａの製造機器ＭＡ１（１３３３）を制御する制御用アプリケーションプログラムＡＰ２（１３０２）を持つサービスエンティティＳＥ２（１３２２）を通してアプリケーションプログラムＡＰ２（１３０２）に製造指示を伝え、次に製造工程Ｂの製造機器ＭＢ１（１３３３）を制御する制御用アプリケーションプログラムＡＰ３（１３０３）を持つサービスエンティティＳＥ３（１３２３）を通してアプリケーションプログラムＡＰ３（１３０３）に製造指示を伝え、その後サービスエージェントＳＡ１（１４００）を作成したサービスエンティティＳＥ１（１３２１）に戻るものとする。

【０１１９】図２３に利用者がサービスエンティティＳＥ１（１３２１）を用いてコンピュータＣ１（１３４１）上で作成した時点の製造指示書を表すサービスエージェントの一例であるサービスエージェントＳＡ１（１４００）の構造を示す。サービスエージェントＳＡ１（１４００）は、識別子ＩＤ１（１４０１）、移動先リスト一覧（（Ｃ２　ＳＥ２）（Ｃ３　ＳＥ３）（Ｃ１　ＳＥ１））（１４０２）、移動リスト一覧ｎｉｌ（１４０３）、移動先名（Ｃ２　ＳＥ２）（１４０４）に加え、全データ項目として、製造機器ＭＡ１の設定パラメータである項目であるＡ工程製造速度項目（１４０５）とその値１０、製造機器ＭＡ１から取得するデータの項目であるＡ工程所要時間項目（１４０６）、製造機器Ｍ

Ｂ１の設定パラメータである項目であるＢ工程製造速度項目（１４０７）とその値１５、製造機器ＭＢ１から取得するデータの項目であるＢ工程所要時間項目（１４０８）を持つ。さらにデータ出力手続きとして、出力データ選択手続き（１４２１）、出力データ変換手続き（１４２２）、パラメタ設定手続き（１４２３）を、データ入力手続きとして、データ取得手続き（１４１１）、入力データ選択手続き（１４１２）、入力データ格納手続き（１４１３）を、さらに制御手続き（１４３０）を持つ。

【０１２０】サービスエージェントＳＥ１（１３２１）を用いて生成されたサービスエージェントＳＡ１（１４００）は、チャネル（４０）を通してコンピュータＣ１（１３４１）のルーティングマネージャＲＭ１（１３１１）に送られる。ルーティングマネージャＲＭ１（１３１１）は、サービスエージェントＳＡ１（１４００）の移動先名の値（Ｃ２　ＳＥ２）（１４０４）をチェックし、移動先名に指定されたコンピュータＣ２（１３４２）との間に接続されたチャネル（４０）を検索し、その結果得られたチャネル（４０）を通してコンピュータＣ２（１３４２）にサービスエージェントＳＡ１（１４００）を送る。

【０１２１】コンピュータＣ２（１３４２）上のルーティングマネージャＲＭ２（１３１２）は、送られてきたサービスエージェントＳＡ１（１４００）を受け取ると、その移動先名の値（Ｃ２　ＳＥ２）（１４０４）をチェックし、移動先名に指定されたサービスエンティティＳＥ２（１３２２）との間のに接続されたチャネル（４０）を検索し、その結果得られたチャネル（４０）を通してサービスエンティティＳＥ２（１３２２）にサービスエージェントＳＡ１（１４００）を送る。サービスエンティティＳＥ２（１３２２）は、チャネル（４０）を通してサービスエージェントＳＡ１（１４００）を受け取ると、サービスエンティティＳＥ２（１３２２）の連携インタフェース（３０）がサービスエージェントＳＡ１（１４００）の持つ出力データ選択手続き（１４２１）を取得し、その手続き（１４２１）を実行し、サービスエンティティＳＥ２（１３２２）の持つアプリケーションプログラムＡＰ２（１３０２）に渡すべきデータ項目であるＡ工程製造速度項目（１４０５）を決定する。次に、サービスエージェントＳＡ１（１４００）の持つ出力データ変換手続き（１４２２）を取得し、その手続き（１４２２）を実行し、選択された項目であるＡ工程製造速度項目（１４０５）の値１０を取り出し、そのデータがアプリケーションプログラムに渡すデータであることを示すタグを付加した後、サービスエージェントＳＡ１（１４００）の持つパラメタ設定手続き（１４２３）を取得し、その手続き（１４２３）を実行し、アプリケーションプログラムＡＰ２（１３０２）に指定された値１０を設定する。次にサービスエージェントＳＡ１（１４００）の持つ制御手続き（１４３０）を取得し、その手続き（１４３０）を実行することによりアプリケーションプログラムＡＰ２（１３０２）の処理を行なう。

【０１２２】アプリケーションプログラムＡＰ２（１３０２）の処理が終了すると、サービスエージェントＳＡ１（１４００）の持つデータ取得手続き（１４１１）を取得し、その手続き（１４１１）を実行し、アプリケーションプログラムの実行結果を取得する。次にサービスエージェントＳＡ１（１４００）の持つ入力データ選択手続き（１４１２）を取得し、その手続き（１４１２）を実行し、アプリケーションプログラムの実行結果を格納するデータ項目であるＡ工程所要時間（１４０６）を選択し、次にサービスエージェントＳＡ１（１４００）の持つ入力データ格納手続き（１４１３）を取得し、その手続き（１４１３）を実行し、アプリケーションプログラムＡＰ２（１３０２）の実行結果をＡ工程所要時間項目（１４０６）に格納する。

【０１２３】次に移動先リスト一覧（１４０２）の２番目の要素である（Ｃ３　ＳＥ３）を取り出し、それを移動先名（１４０４）に格納し、チャネル（４０）を通してコンピュータＣ２（１３４２）上のルーティングマネージャＲＭ２（１３１２）に送る。

【０１２４】ルーティングマネージャＲＭ２（１３１２）は、サービスエンティティＳＥ２（１３２２）からチャネル（４０）を通して送られてきたサービスエージェントＳＡ１（１４００）を受け取ると、その移動先名（Ｃ３　ＳＥ３）（１４０４）をチェックし、移動先名（Ｃ３　ＳＥ３）（１４０４）に指定されたコンピュータＣ３（１３４３）に接続されたチャネル（４０）を検索し、その結果得られたチャネル（４０）を通してサービスエージェントＳＡ１（１４００）を送る。

【０１２５】コンピュータＣ３（１３４３）上のルーティングマネージャＲＭ３（１３１３）は、送られてきたサービスエージェントＳＡ１（１４００）を受け取ると、その移動先名の値（Ｃ３　ＳＥ３）（１４０４）をチェックし、移動先名（１４０４）に指定されたサービスエンティティＳＥ３（１３２３）との間に接続されたチャネル（４０）を検索し、その結果得られたチャネル（４０）を通してサービスエンティティＳＥ３（１３２３）にサービスエージェントＳＡ１（１４００）を送る。

【０１２６】サービスエンティティＳＥ３（１３２３）は、チャネル（４０）を通してサービスエージェントＳＡ１（１４００）を受け取ると、サービスエンティティＳＥ３（１３２３）の連携インタフェース（３０）がサービスエージェントＳＡ１（１４００）の持つ出力データ選択手続き（１４２１）を取得し、その手続き（１４２１）を実行し、サービスエンティティＳＥ３（１３２３）の持つアプリケーションプログラムＡＰ３（１３０

3）に渡すべきデータ項目であるB工程製造速度項目（１４０７）を決定する。次に、サービスエージェントSA1（１４００）の持つ出力データ変換手続き（１４２２）を取得し、その手続き（１４２２）を実行し、選択された項目であるB工程製造速度（１４０７）の値１５を取り出し、そのデータがアプリケーションプログラムに渡すデータであることを示すタグを付加した後、サービスエージェントSA1（１４００）の持つパラメタ設定手続き（１４２３）を取得し、その手続き（１４２３）を実行し、アプリケーションプログラムAP3（１３０３）に指定された値１５を設定する。次にサービスエージェントSA1（１４００）の持つ制御手続き（１４３０）を取得し、その手続き（１４３０）を実行することによりアプリケーションプログラムAP3（１３０３）の処理を行なう。

【０１２７】アプリケーションプログラムAP3（１３０３）の処理が終了すると、サービスエージェントSA1（１４００）の持つデータ取得手続き（１４１１）を取得し、その手続き（１４１１）を実行し、アプリケーションプログラムの実行結果を取得する。次にサービスエージェントSA1（１４００）の持つ入力データ選択手続き（１４１２）を取得し、その手続き（１４１２）を実行し、アプリケーションプログラムの実行結果を格納するデータ項目であるB工程所要時間項目（１４０８）を選択し、次にサービスエージェントSA1（１４００）の持つ入力データ格納手続き（１４１３）を取得し、その手続き（１４１３）を実行し、アプリケーションプログラムAP3（１３０３）の実行結果をB工程所要時間項目（１４０８）に格納する。

【０１２８】次に移動先リスト一覧（１４０２）の3番目の要素である（C1　SE1）を取り出し、それを移動先名（１４０４）に格納し、チャネル（４０）を通してコンピュータC3（１３４３）上のルーティングマネージャRM3（１３１３）に送る。

【０１２９】ルーティングマネージャRM3（１３１３）は、サービスエンティティSE3（１３２３）からチャネル（４０）を通して送られてきたサービスエージェントSA1（１４００）を受け取ると、その移動先名（C1　SE1）（１４０４）をチェックし、移動先名（C1　SE1）（１４０４）に指定されたコンピュータC1（１３４１）に接続されたチャネル（４０）を検索するが、コンピュータC3（１３４３）上のルーティングマネージャRM3（１３１３）と接続されているコンピュータの中にはコンピュータ名C1（１３４１）がないため、接続されているコンピュータC2（１３４２）のルーティングマネージャRM2（１３１２）にサービスエージェントSA1（１４００）を送る。

【０１３０】コンピュータC2（１３４２）上のルーティングマネージャRM2（１３１２）は、送られてきたサービスエージェントSA1（１４００）を受け取ると、その移動先名の値（C1　SE1）（１４０４）をチェックし、移動先名（１４０４）に指定されたコンピュータ名C1がコンピュータ名C2と異なるため、サービスエージェントSA1（１４００）の移動リスト（１４０３）にコンピュータ名C2を格納し、移動リスト（１４０３）の値を（（C2））とし、サービスエージェントSA1（１４００）の移動先名（C1　SE1）（１４０４）に指定されたコンピュータ名C1と接続されたチャネル（４０）を検索し、その結果得られたチャネル（４０）にサービスエージェントSA1（１４００）を送る。

【０１３１】ルーティングマネージャRM1（１３１１）は、ルーティングマネージャRM2（１３１２）からチャネル（４０）を通して送られてきたサービスエージェントSA1（１４００）を受け取ると、その移動先名（C1　SE1）（１４０４）をチェックし、移動先名（C1　SE1）（１４０４）とコンピュータ名C1が同一であることをチェックした後、移動先名（C1　SE1）（１４０４）に指定されたサービスエンティティSE1（１３２１）との間に接続されたチャネル（４０）を検索し、その結果得られたチャネル（４０）を通してサービスエンティティSE1（１３２１）にサービスエージェントSA1（１４００）を送る。

【０１３２】このようにしてサービスエージェントSA1（１４００）を用いることにより、製造工程A、Bを管理するコンピュータC2（１３４２）、C3（１３４３）に接続された製造機器MA1（１３３２）、MB1（１３３３）を制御するアプリケーションプログラムAP2（１３０２）、AP3（１３０３）を連携して制御することが可能となる。また、製造工程が異なる場合でも、製造指示書を作成する際に、その移動先リスト一覧（１４０２）に指定する移動先の一覧を変更するだけで制御する製造装置やその順序を変更することができ、容易に連携の方法を変更することができる。

【０１３３】

【発明の効果】以上のように、本発明では、さまざまなアプリケーションプログラムを柔軟に連携処理するために、アプリケーションプログラム本体に連携処理のための連携インタフェースを付与し、連携のために必要な連携処理手続きと連携する際に必要なデータ、及びデータの格納項目、個々のアプリケーションプログラム本体の実行の制御手続きを作業指示書に記述し、その作業指示書をコンピュータ間で移動させることにより、柔軟なアプリケーションプログラム間の連携処理が可能となる。

【０１３４】これにより、連携処理の対象とするアプリケーションプログラムの実行を停止することなく、アプリケーションプログラム間の連携処理手続きを作業指示書により変更することが可能となる。また、利用者は連携処理をしたいアプリケーションプログラムが分散処理システム上のどのコンピュータで動作しているかを意識

39

することなく、利用したいアプリケーションプログラム本体（サービスエンティティ名）を指定することで、そのアプリケーションプログラムを利用することができる。

【０１３５】また、複数の作業者に依頼し、各作業者にアプリケーションプログラムを操作して行ってもらっていた一連の作業を自動化することができる。

【図面の簡単な説明】

【図１】本発明によるアプリケーションプログラム間連携処理を行う分散処理システムの実施例の構成を示す構成図である。

【図２】図１に示したコンピュータの構成を示す構成図である。

【図３】実施例におけるサービスエージェントの構造を示す説明図である。

【図４】実施例ににおけるルーティングマネージャ間の通信路の接続状態の説明図である。

【図５】図４の接続状態に対して新たなルーティングマネージャが接続された状態を示す説明図である。

【図６】実施例におけるルーティングマネージャとサービスエンティティの接続状態の説明図である。

【図７】実施例におけるルーティングマネージャのプログラム構造を示すブロック図である。

【図８】図７に示したテーブル類の構造の説明図である。

【図９】実施例における連携インタフェース３０のプログラム構造を示すブロック図である。示した

【図１０】図９に示した関数テーブル３７１の構造の説明図である。

【図１１】図３に示した移動先リスト一覧２０２の移動

40

先リスト９１の記述形式および記述例の説明図である。

【図１２】図３に示した移動リスト一覧２０３の移動リスト１０１の記述方法および記述例の説明図である。

【図１３】実施例におけるサービスエージェントの移動処理手続きを示すフローチャート（その１）を示す。

【図１４】実施例におけるサービスエージェントの移動処理手続きを示すフローチャート（その２）を示す。

【図１５】実施例におけるサービスエージェントの移動処理手続きを示すフローチャート（その３）を示す。

【図１６】実施例におけるサービスエージェントの移動処理手続きを示すフローチャート（その４）を示す。

【図１７】実施例における連携インタフェース３０の処理を示すフローチャート（その１）である。

【図１８】実施例における連携インタフェース３０の処理を示すフローチャート（その２）である。

【図１９】実施例における連携インタフェース３０の処理を示すフローチャート（その３）である。

【図２０】実施例における連携インタフェース３０の処理を示すフローチャート（その４）である。

【図２１】実施例におけるデータ項目の組み合わせによる移動先決定方法の例の説明図である。

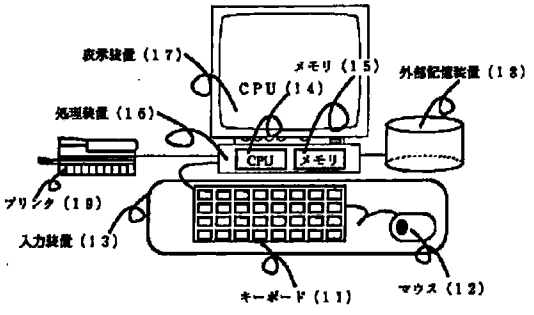
【図２２】本発明を適用した製造ライン管理システムの構成例を示す構成図である。

【図２３】図２２のシステムにおけるサービスエージェントの構造例を示す説明図である。

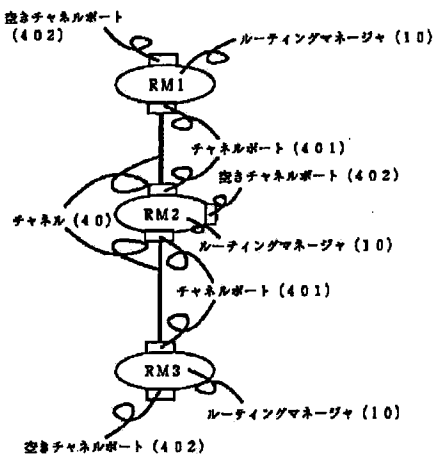
【符号の説明】

１…コンピュータ、３…アプリケーションプログラム本体、４…サービスエンティティ、１０…ルーティングマネージャ、２０…サービスエージェント、３０…連携インタフェース。

【図２】

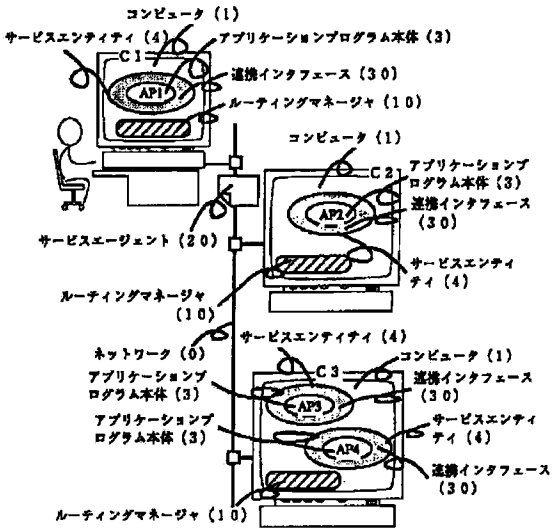


【図４】



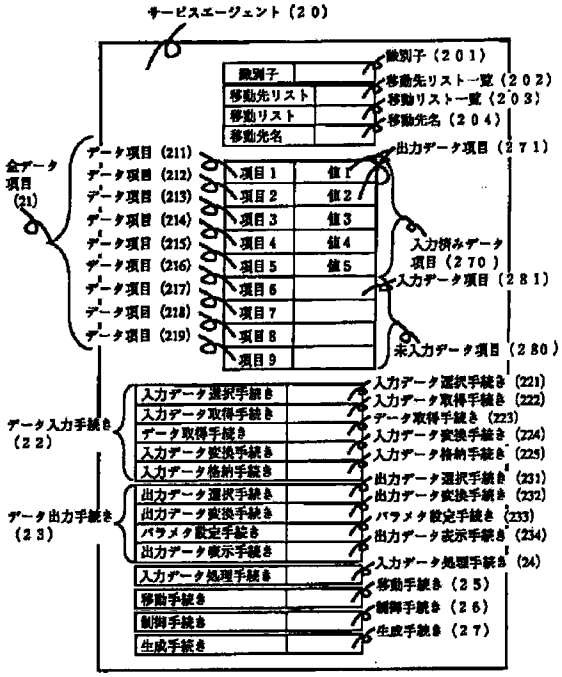
【図１】

図１　アプリケーションプログラム間連携処理システム構成



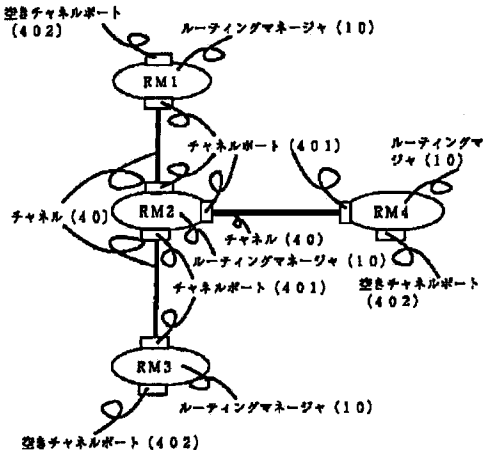
【図３】

図３　サービスエージェントの構造



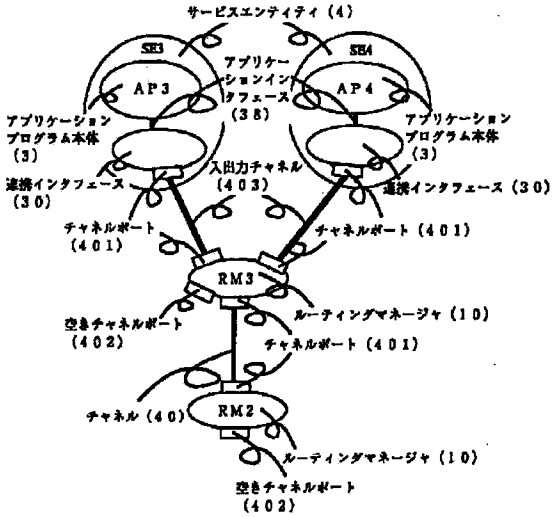
【図５】

図５　新たにルーティングマネージャが接続された状態



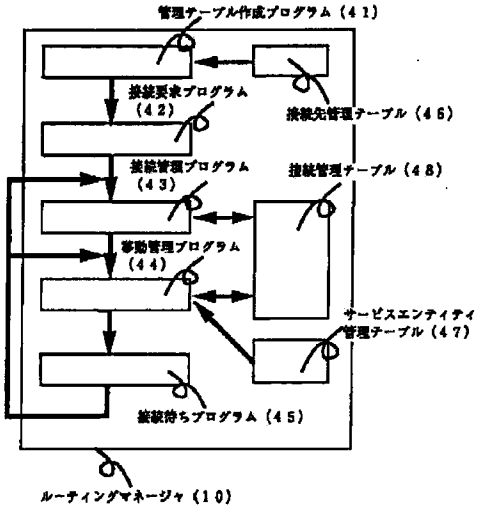
【図６】

図６　ルーティングマネージャとサービスエンティティの接続状態



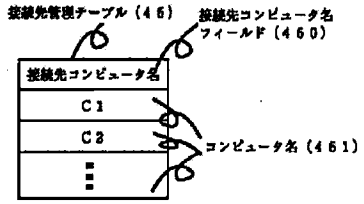
【図７】

図７　ルーティングマネージャのプログラム構造

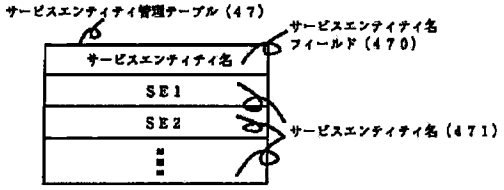


【図８】

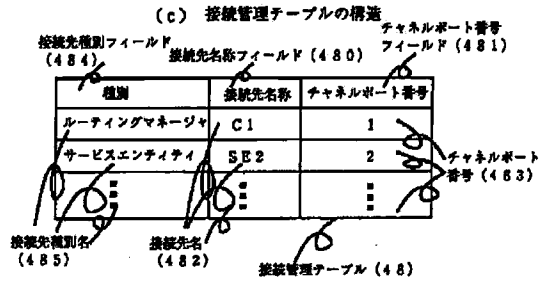
図８　（ａ）接続先管理テーブルの構造



（ｂ）サービスエンティティ管理テーブルの構造

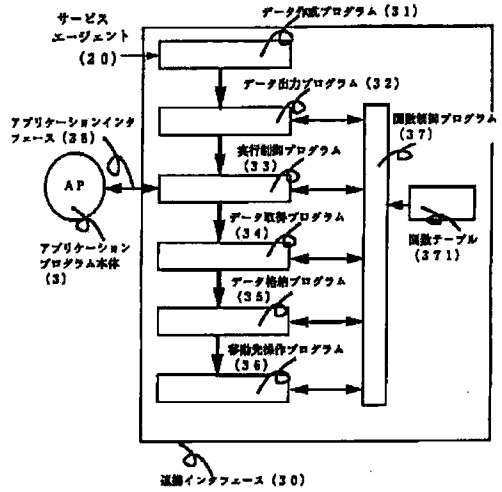


（ｃ）接続管理テーブルの構造



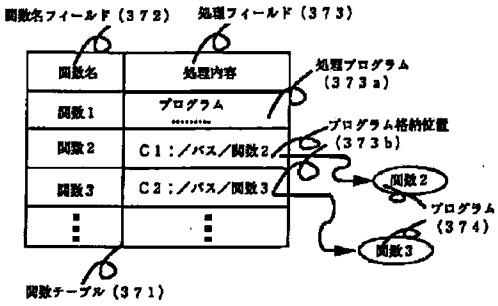
【図９】

図９　連携インタフェースのプログラム構造

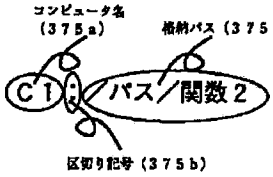


【図１０】

図１０　（ａ）関数テーブルの構造

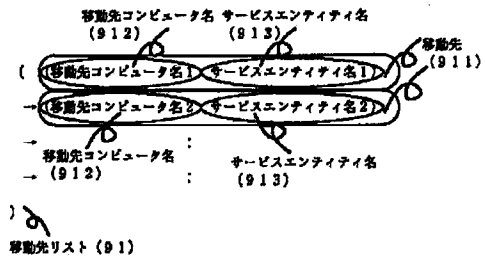


（ｂ）プログラム格納位置の指定方法

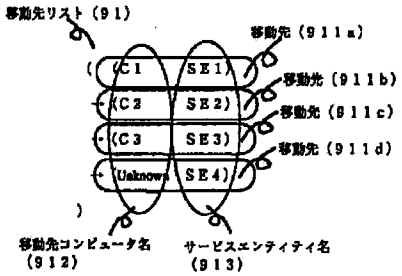


【図11】

図11　(a) 移動先リストの記述形式

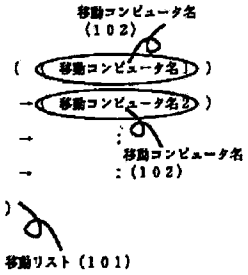


(b) 移動先リストの記述例

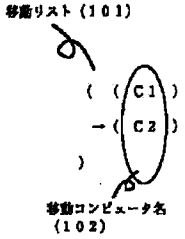


【図12】

図12　(a) 移動リストの記述方法

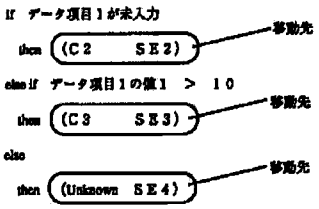


(b) 移動リストの記述例



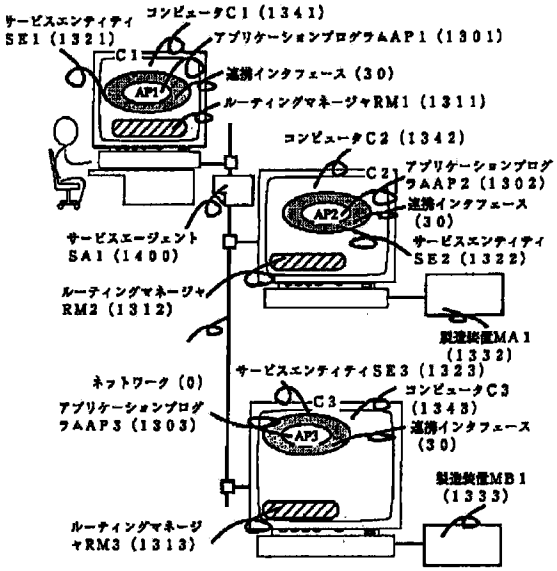
【図21】

図21　データ項目の組合せによる移動先決定方法の例



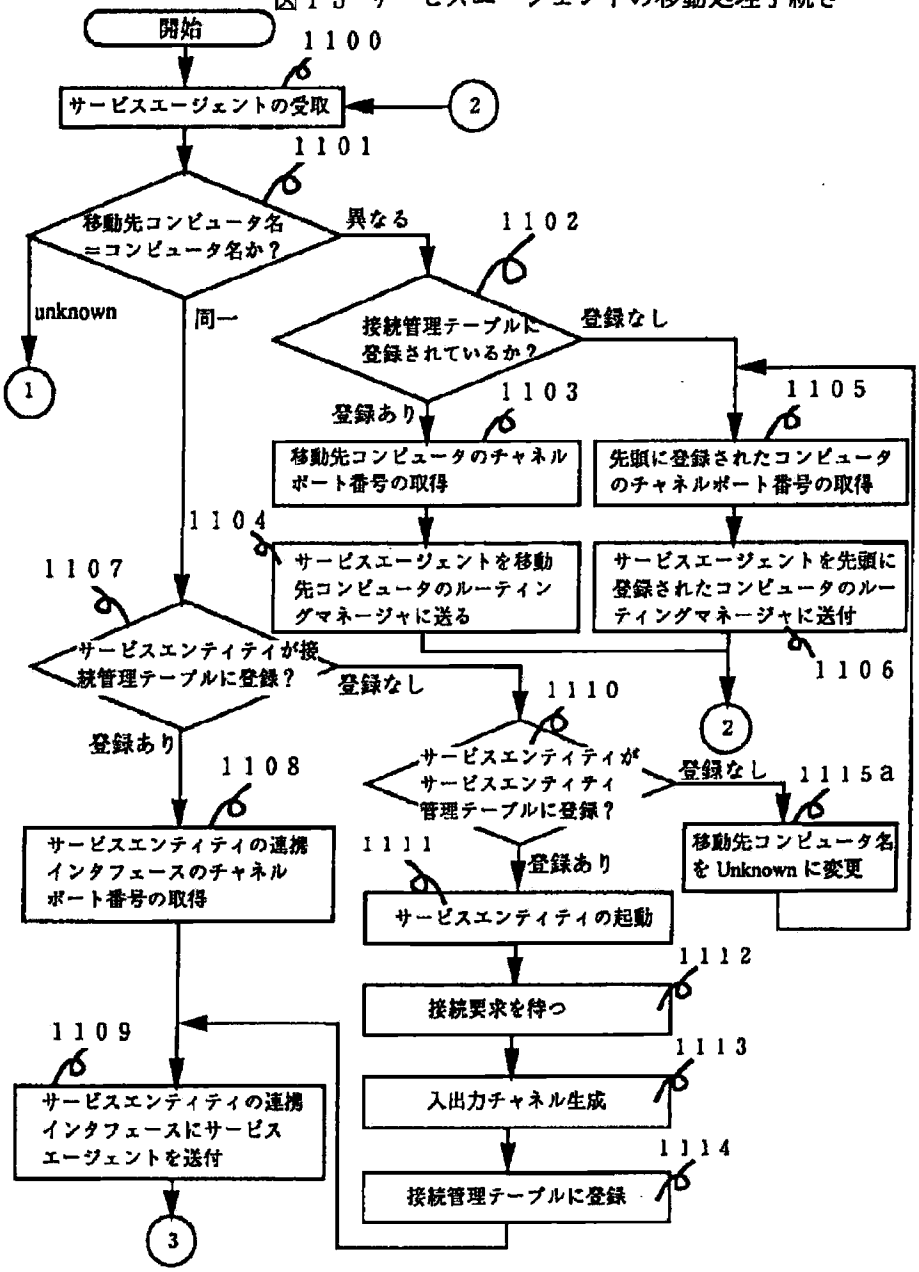
【図22】

図22　本発明による製造ライン管理システムの例



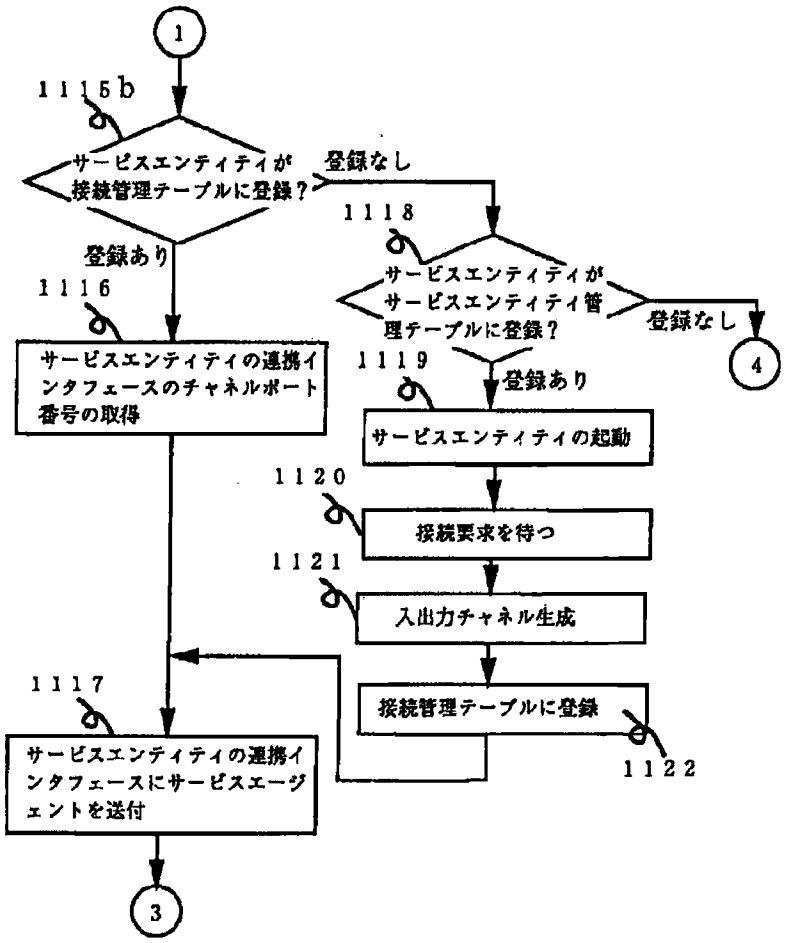
【図１３】

図１３ サービスエージェントの移動処理手続き



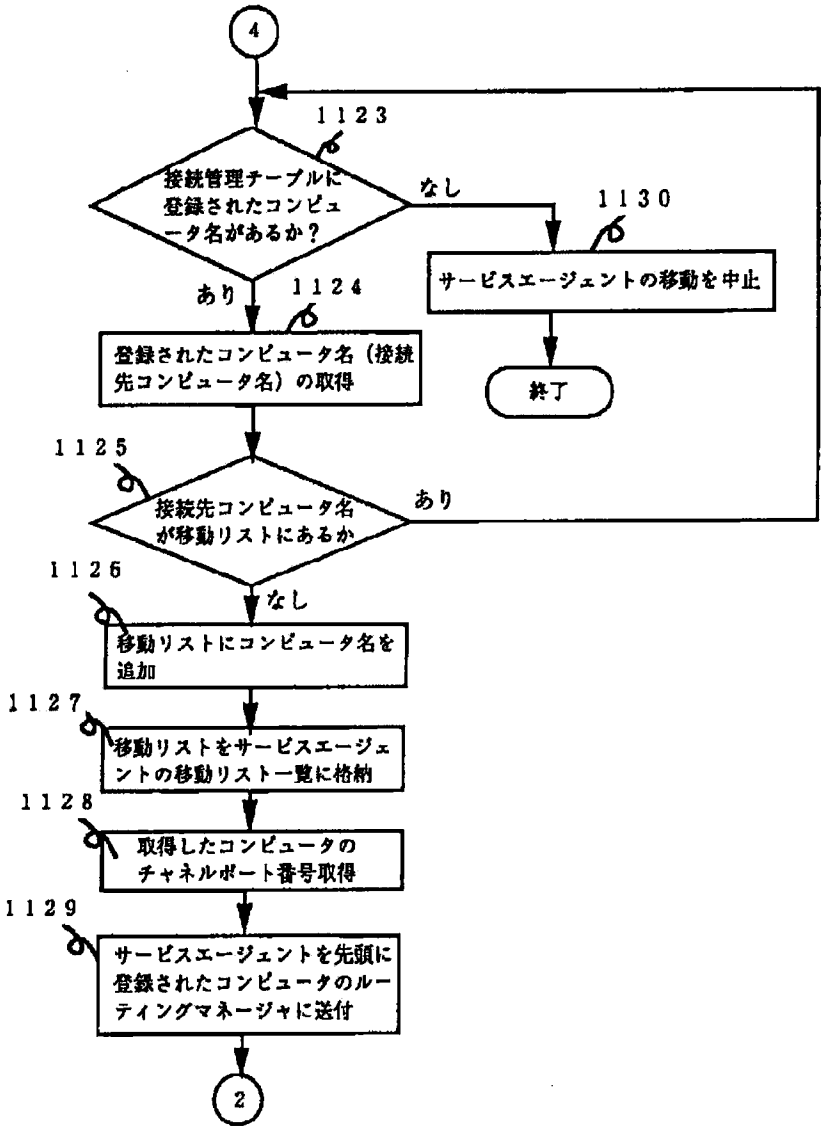
【図１４】

図１４　サービスエージェントの移動処理手続き



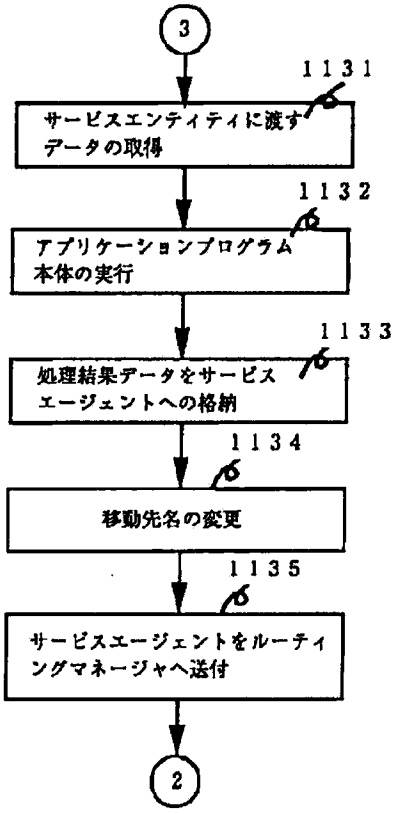
【図１５】

図１５　サービスエージェントの移動処理手続き



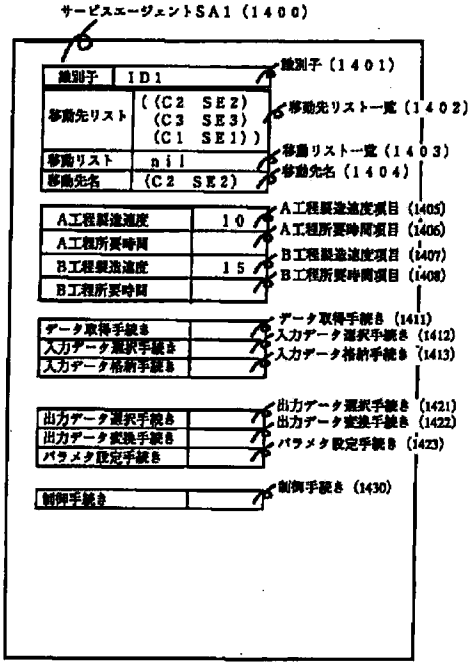
【図１６】

図１６　サービスエージェントの移動処理手続き



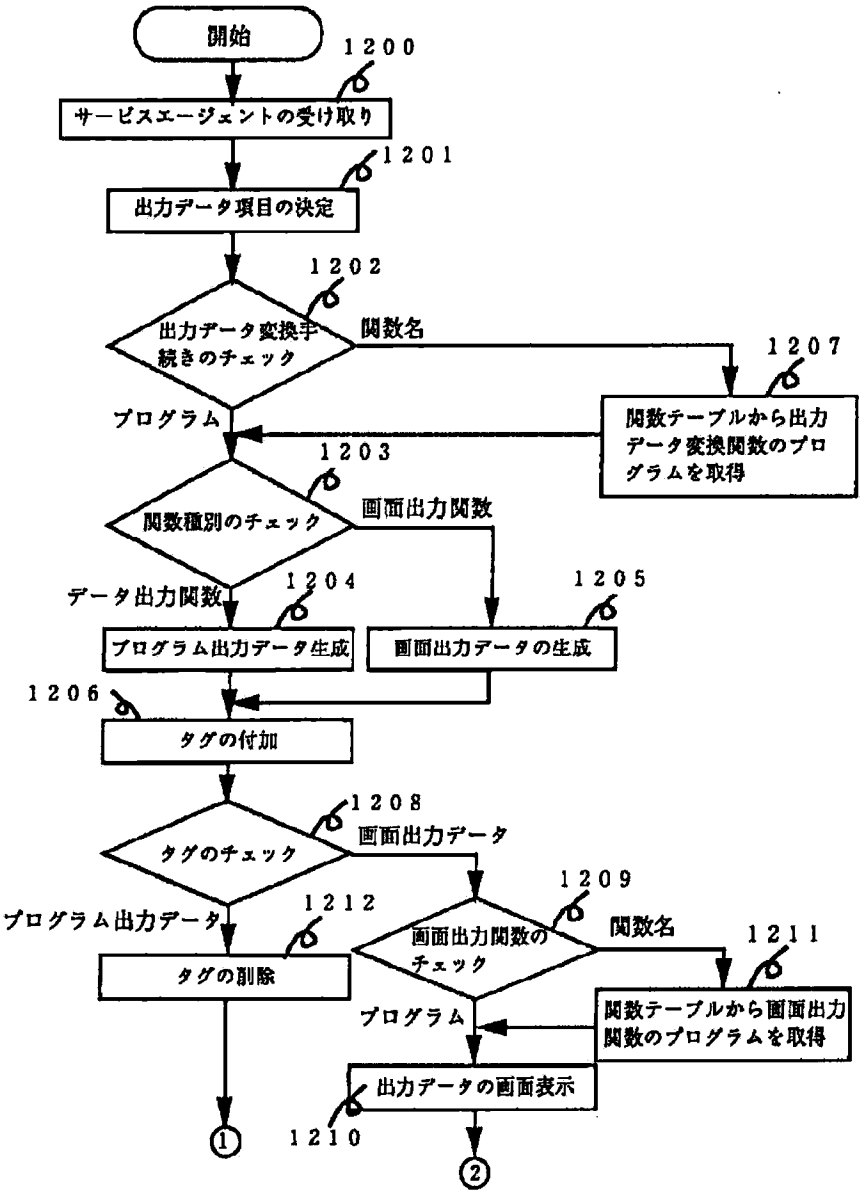
【図２３】

図２３　サービスエージェントＳＡ１の構造



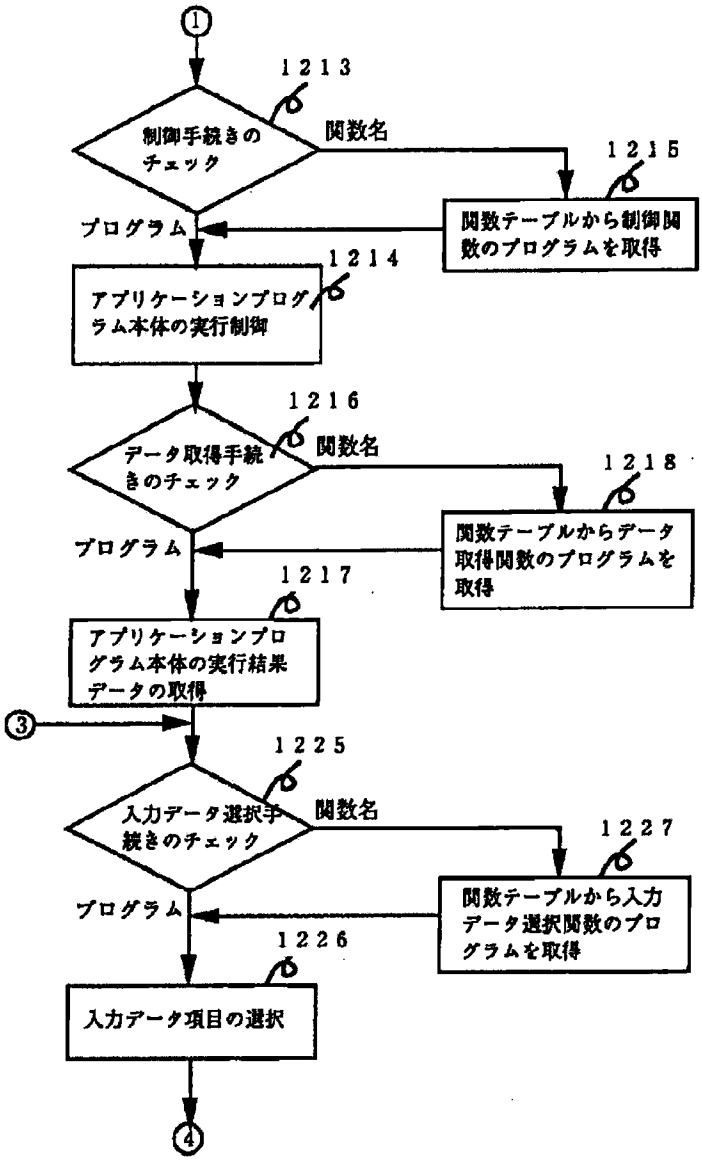
【図１７】

図１７　連携インタフェースの処理フロー



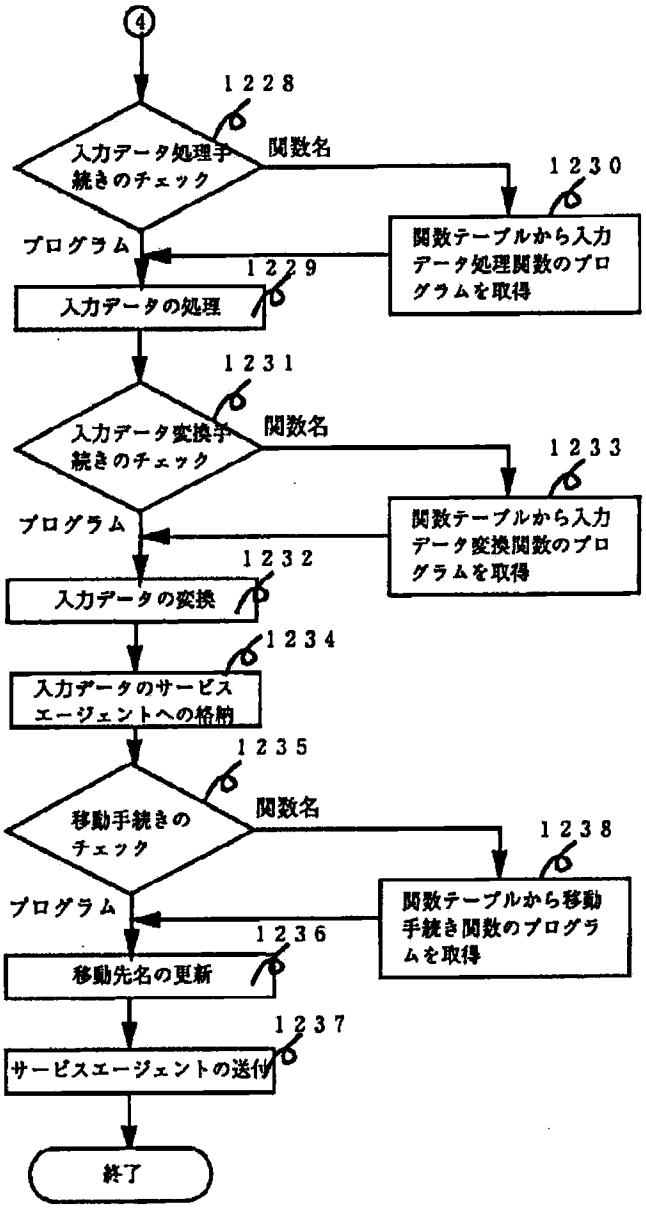
【図１８】

図１８　連携インタフェースの処理フロー



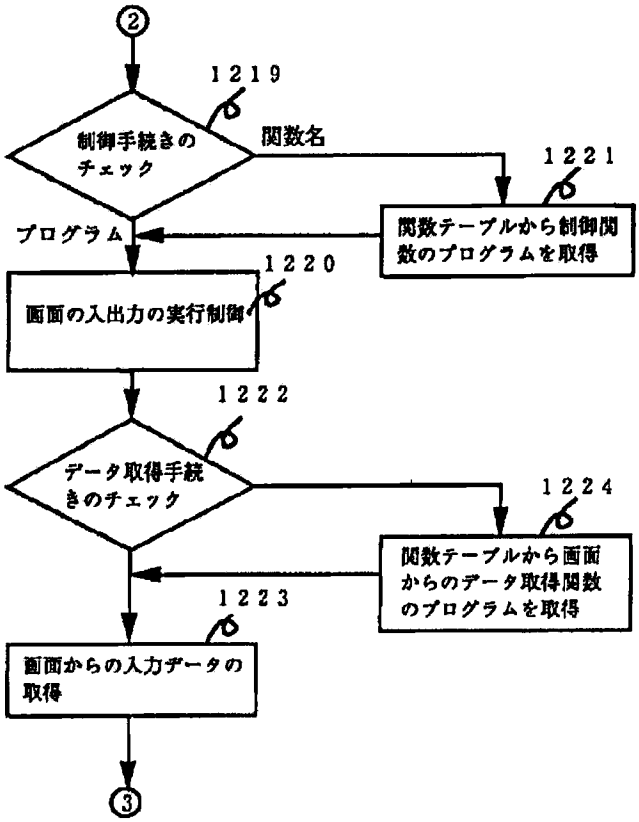
【図19】

図19　連携インタフェースの処理フロー



【図20】

図20　連携インタフェースの処理フロー



フロントページの続き

(72)発明者　高橋　勉
宮城県仙台市青葉区一番町二丁目4番1号
日立東北ソフトウェア株式会社内

PTO 08-0614

CC=JP
DATE=19961129
KIND=A
PN=08314872

COOPERATIVE PROCESSING METHOD AMONG APPLICATION PROGRAMS

[Apurikeshon Puroguramu Kan Renkei Shori Hoho]

Toshiaki ITO, et al.

UNITED STATES PATENT AND TRADEMARK OFFICE
WASHINGTON, D.C. NOVEMBER 2007
TRANSLATED BY: SCHREIBER TRANSLATION, INC.

PUBLICATION COUNTRY	(10):	JP
DOCUMENT NUMBER	(11):	08314872
DOCUMENT KIND	(12):	A
PUBLICATION DATE	(43):	19961129
APPLICATION NUMBER	(21):	07138641
APPLICATION DATE	(22):	19950512
INTERNATIONAL CLASSIFICATION	(51):	G06F 15/16 9/46 15/00
PRIORITY COUNTRY	(33):	N/A
PRIORITY NUMBER	(31):	N/A
PRIORITY DATE	(32):	N/A
INVENTOR(S)	(72):	Toshiaki ITO, et al.
APPLICANT(S)	(71):	Hitachi Tohoku Software Co., Ltd.
DESIGNATED CONTRACTING STATES	(81):	N/A
TITLE	(54):	COOPERATIVE PROCESSING METHOD AMONG APPLICATION PROGRAMS
FOREIGN TITLE	(54A):	Apurikeshon Puroguramu Kan Renkei Shori Hoho

/1[1]

SPECIFICATION

(54) [Title of the Invention]

Cooperative Processing Method among Application Programs

/2

[Claims]

[Claim 1] A cooperative processing method among application programs which is a cooperative processing method among plural application programs operating on plural computers connected to a computer network and

is characterized in that

an operation instruction describing utilization orders of the plural application programs and utilization procedures of respective application programs is prepared,

the operation instruction is successively transferred to the plural computers having objective application programs through the computer network,

a cooperative interface cooperating with the application programs in accordance with the utilization procedures of the application programs described in the operation instruction are provided in each of the plural application programs, when each

[1]Numbers in the margin indicate pagination in the foreign text.

cooperative interface receives the operation instruction, the cooperative interface starts up its own application programs in accordance with the utilization procedures relating to its own application programs described in the operation instruction and performs an objective control and/or giving and receiving of data, and

a series of operations based on the plural application programs are executed by patrolling computers having necessary application programs.

[Claim 2] The cooperative processing method among application programs according to Claim 1, which is characterized by comprising at least one of an all data item for indicating a summary of data items used in all application programs utilized by the operation instruction, an output data item for storing data to be given to each application program, an input data item for storing processing result of each application program, a data output procedure for indicating a procedure outputting the data stored in the output data item to the application program, a data input procedure for indicating an input procedure of data of each input data item, an input data processing procedure for processing the data inputted by the data input procedure and a

control procedure for controlling the execution of the application programs as constituents of the utilization procedures described in the operation instruction.

[Claim 3] The cooperative processing method among application programs according to Claim 1 or 2, which is characterized in that the method has a movement control means on each computer, and the movement control means ensures and manages a communication path between the cooperative interface on its own computer and other computers and controls the movement of the operation instruction according to the utilization procedures described in the operation instruction received through the communication path.

[Claim 4] The cooperative processing method among application programs according to Claim 2, which is characterized by having at least one of a start procedure for starting the application programs of the cooperative interface receiving the operation instruction, an end procedure for ending the execution of application programs, an execution procedure for executing one or more commands held by the application programs and a control script combining these procedures as control procedures that are constituents of the operation instruction.

[Claim 5] The cooperative processing method among application programs according to Claim 2 which is characterized by

having a generation procedure for generating a new operation instruction as a constituent of the operation instruction.

[Claim 6] The cooperative processing method among application programs according to Claim 1 or 2 which is characterized in that the utilization procedures are any of procedures or function names and, in the case of function name, acquire programs corresponding to the function name in the cooperative interface.

[Claim 7] The cooperative processing method among application programs according to Claim 1, 2, 3, 4, 5 or 6 which is characterized in that the utilization orders described in the operation instruction assigns a set of computer names and service entities comprising application programs operating on the computers and cooperative interfaces according to the orders of utilization of the service entities and can omit the computer names.

[Detailed Description of the Invention]

[0001]

[Field of Industrial Application] The present invention relates to a cooperative processing method among application programs which is involved in a dispersion processing system wherein a user performs processing by utilizing plural application

programs while successively combining the programs and, as necessary, can cooperate arrangements and processing among various application programs utilized by operators in performing business constructed from a series of advancing operations while cooperating the mutual operations by exchanging processing courses, processing results and processing contents with operators performing other operations and further easily change cooperative procedures of processing among individual application programs according to business.

/2 ~ 3

[0002] The present invention particularly relates to a utilization technique for flexibly combining and cooperating application programs according to operations to reduce operation load of each operator and smoothen the flow of operations and a dispersion processing system for realizing such a flexible cooperative processing method of application programs by automatizing operation procedures combining plural application programs performed by the users, operations performed at the request of each operator and acquisition of the results and systematizing utilization procedures of application programs necessary for a series of operations.

[0003]

[Prior Art] A dispersion processing system is a system wherein computers used in operations to be performed by individual operators are connected by a network to exchange data needed among individual operators. Such a system are individual utilization centers wherein one by one operations are performed by the operators with the application programs, only data needed among the operators are shared, and the data shared by the application programs and utilized by the operators can be utilized.

[0004] The construction for cooperatively operating application programs on a dispersion processing system of a four cooperative processing mode (RPC (Remote Procedure Call), conversation, file transfer and message-base) has been discussed in *Nikkei Computers:* ⌈Non-Synchronization and Accumulation type Communication Eliminate Weak Points of Client Servers⌋, 159 ~ 168 (Aug. 8, 1988) and *Nikkei Computers:* ⌈Message-Base Middleware for Easily Realizing Non-Synchronization Cooperation of Dispersion Applications with Each Other Rapidly Increases⌋, 67 ~ 74 (Sep. 20, 1993).

[0005] ⌈RPC⌋ mode is a mode wherein one module of application programs is fetched and the module is executed by other computers. ⌈Conversation⌋ mode is a mode wherein data or processing requests are exchanged among plural application programs

through a real-time communication path by connecting the programs with the real-time communication path. A system wherein application programs used in operations are divided into a processing request source (client) and a processing execution source (server) to perform cooperative processing among application programs on the client side and the server side is a client-server system. In these two modes, a cooperative processing for performing processing while taking synchronization among application programs on the client-side and the server side can be realized by promptly processing a processing request from the transmission-side application programs if the request is received and returning a processing result to the transmission-side application programs in a state that the reception-side application programs always operate and wait for a processing request from the transmission-side application programs. In these modes, cooperative procedures among the application programs for performing the cooperative processing are predetermined at the time of developing the application programs, and the cooperative procedures must be described as a program.

[0006] 「File transfer」 is a mode wherein a file is transferred among the cooperated application programs and the application programs perform processing for data described in the file when the data is received. In this mode, a cooperative

processing among non-synchronous application programs can be realized by executing the application programs with the reception of file as a trigger. ⌈Message-base⌋ is a mode wherein a program realizing a message-base processing receives a message sent from the transmission-side application programs, sends the message to the reception-side application programs described in the message, and a cooperative processing among the application programs is realized by executing a processing procedure corresponding to the message in the reception-side application programs receiving the message. In the message-base mode, the format of message is predetermined among the application programs performing the cooperative processing and the processing of application programs is described when the message is received.

[0007] In these four cooperative processing modes for application programs, a processing procedure necessary for cooperation must be determined for each application program and described as a program to perform business constructed from a series of operations by cooperating processing among plural application programs used by individual operations.

[0008] In logic communication path control modes among programs described in Japan Kokai 02-186737, a mode wherein a cooperative processing among application programs is realized by exchanging data among application programs operating on com-

puters connected to a network. In this mode, the cooperative processing among application programs is realized based on sharing display of pictures among the application programs by exchanging data of operations (change, supplement, deletion, etc. of data) applied by users via pictures of the application programs among the same application programs through a logic communication path.

[0009] In the cooperative processing mode among these application programs, when data and messages used in the cooperative processing are sent to application programs on other computers, the data and messages can be sent to the application programs on assigned computers by assigning sending destinations of the data and messages by any one method for name of destination computers, physical positions or physical names of destination computers.

[0010] More recently, the research and development of systems for supporting operations straddling plural operators from a system of individual utilization centers based on individual

/4

operators have been prosperously carried out. About the kinds of systems supporting these operations straddling plural operators and the contents of supported operations, systems supporting cooperative operations straddling plural persons researched and developed domestically and abroad have been discussed in H.

Ishii: 「Trend in Technical Research of Groupware」, *J. Dissertation of Information Processing Society of Japan*, 30 (12) (Dec. 1989), H. Ishii: 「Trend in Research of Groupware Support using Computers」, *Computer Software*, 8 (2), 14 ~ 26 (1985), C.A. Ellis, S.J. Gibbs and G.L. Rein: "Groupware: Some Issues and Experiences", CACM 34 (1), 38 ~ 58 (Jan. 1991).

[0011] The system supporting these cooperative operations is a system supporting that each operator cooperatively performs operations of same contents by use of same application program, it displays processing results of the application programs among the operators and provides a function sharing a user interface picture added with some operations to the processing results.

[0012]

[Problem to Be Solved by the Invention] In the above prior art, the cooperative processing among application programs is performed by predetermining data and messages exchanging among some specific application programs, describing processing programs corresponding to the predetermined data and messages in respective application programs and then exchanging the predetermined data and messages among the application programs. Therefore, procedures for the cooperative processing among application programs cannot be changed. Or even they can be changed,

the processing programs in all the application programs relating to the changes must be changed. As a result, it becomes very difficult to change cooperative processing programs among the application programs corresponding to operation contents and objects.

[0013] Moreover, it is necessary to end the execution of all application programs as objects of change to change the cooperative processing procedures among the application programs. Once application programs are used so that various users perform respective operations on a dispersion processing system and end the execution of all application programs as targets of change, the difficulty increases with increasing the number of application programs as targets of change.

[0014] Furthermore, those exchanged for performing the cooperative processing among the application programs are data or messages, therefore processing procedures at the time of receiving these data and messages is determined by the reception-side application programs receiving the data and messages, and it is impossible that transmission-side application programs changes the processing procedures of reception-side application programs. In order to realize the execution of a series of processing procedures combining plural functions provided by the reception-side application programs, i.e., combinations of plural commands to be executed at a time by the reception-side

application programs, a message for executing necessary processing procedures while taking synchronization is sent to the application programs, a pertinent procedure is executed and the execution results are combined by the transmission-side application programs, or a special message for executing plural commands must be described as a program in the reception-side application programs beforehand, thus the development efficiency lowers.

[0015] Thus, it is necessary to describe the processing procedures among the application programs beforehand, therefore it is difficult to describe the processing procedures among the application programs suited to methods utilized by individual user themselves, and it is also difficult to cooperatively processed among the application programs except that cooperative processing procedures among the application programs provided by a system are utilized.

[0016] Moreover, in order to send these data or messages to the application programs, computers in which the application programs operate must be clearly assigned by a means for univocally identifying the computers, such as names or IP addresses of the computers, etc., when computers in which the application programs operate for performing some processing are changed for some reasons, a section in which computers in the cooperative

processing procedures among application programs utilizing the pertinent application program must be changed. Therefore, the prior art is deficient in flexibility for changes of computers or network, application programs constituting a dispersion processing system.

[0017] One purpose of the present invention consists in providing a cooperative processing method among application programs which enables to perform a cooperative processing among application programs suited to a method utilized by individual user themselves.

[0018] Another purpose of the present invention consists in providing a cooperative processing method among application programs which can flexibly make changes of cooperative processing procedures among application programs without lowering the developing efficiency by giving a cooperative interface capable of giving/receiving the cooperative processing procedures among application programs in addition to data and messages, processing procedures performed in reception-side application programs by receiving the data and messages and a series of processing procedures combining plural functions provided by the reception-side application programs and executing these received procedures.

[0019]

/5

[Means for Solving the Problem] To achieve the above purposes, the present invention is a cooperative processing method among application programs operating on plural computers connected to a computer network wherein

an operation instruction describing utilization orders of the plural application programs and utilization procedures of respective application programs is prepared,

the operation instruction is successively transferred to the plural computer having objective application programs through the computer network,

a cooperative interface cooperating with the application programs in accordance with the utilization procedures of the application programs described in the operation instruction are provided in each of the plural application programs, when each cooperative interface receives the operation instruction, the cooperative interface starts its own application program in accordance with the utilization procedures relating to its own application programs described in the operation instruction and performs an objective control and/or giving and receiving of data, and

a series of operations based on the plural application programs are executed by patrolling computers having necessary application programs.

[0020] This cooperative processing method among application programs comprises at least one of an all data item for indicating a summary of data items used in all application programs utilized by the operation instruction, an output data item for storing data to be given to each application program, an input data item for storing processing result of each application program, a data output procedure for indicating a procedure outputting the data stored in the output data item to the application program, a data input procedure for indicating an input procedure of data of each input data item, an input data processing procedure for processing the data inputted by the data input procedure and a control procedure for controlling the execution of the application programs as constituents of the utilization procedures described in the operation instruction.

[0021] The present invention preferably has a moving control means on each computer, and the moving control means ensures and manages a communication path between the cooperative interface on its own computer and other computers and controls the movement of the operation instruction according to the utiliza-

tion procedures described in the operation instruction received through the communication path.

[0022] For example, the present invention has at least one of a start procedure for starting the application programs of the cooperative interface receiving the operation instruction, an end procedure for ending the execution of application programs, an execution procedure for executing one or more commands having the application programs and a control script combining these procedures as control procedures that are constituents of the operation instruction.

[0023] The present invention may has a generation procedure for generating a new operation instruction as a constituent of the operation instruction. For example, the utilization procedures are any of procedures or function names and, in the case of function names, the programs corresponding to the function names are acquired in the cooperative interface.

[0024] The utilization orders described in the operation instruction assigns a set of computer names and names of service entities comprising application programs operating on the computers and cooperative interfaces according to utilization orders of the service entities and can omit the computer names.

[0025] More specifically, the cooperative processing among application programs in the present invention is constituted by

an operating instruction (service agent) having service entities for giving cooperative interfaces executing cooperative processing procedures among application programs in individual application program bodies involved in the cooperative processing among application programs, utilization orders for indicating movement destinations among the cooperated service entities and utilization procedures of the service entities, and movement control means (routine managers) for generating, maintaining and managing a communication path on which these service entities are moved to the movement destinations described in the utilization orders having the entities.

[0026] The service entities are arranged and executed on individual computers constituting a dispersion processing system as execution units of one application program. The cooperative interfaces constructing the service entities select control procedures possessed by the service agent, input/output data procedures and procedures pertinent to current service entities from display procedures and have functions of executing and controlling the application program bodies by executing the procedures. When a program corresponding to the executing procedures is on other computers, the cooperative interfaces have functions of copying the program on the computers operated by

the service entities and executing procedures required by the service entities.

[0027] The service entities for realizing the cooperative processing among application programs have utilization orders for describing orders cooperating the application programs in the cooperative processing as utilization orders of the service entities having pertinent application programs, and utilization procedures which comprises a control procedure for utilizing functions held by the application programs used in respective service entities, an input/output data processing procedure for transferring data items storing data necessary for actually performing the processing in application programs held by the service entities and data stored in the data items to application programs by the control procedures and storing the processing

/6

result data in the pertinent data items, and a display procedure for displaying data held by the service agent through graphical user interfaces.

[0028] The routine managers are arranged and operated one by one on the computers constituting the dispersion processing system, and have functions of constructing a communication path for moving the service agent among the routine managers operating on the computers and managing the communication path for

moving the service agent among names of service entities operating on the computers and the service entities. Moreover, they have functions of judging whether service entities required by the service agent and received from other routine managers through the communication path are available on their own computers, sending the service agent to the service entities requiring them if they are available or moving the service agent to routine managers on other computers if they are unavailable. The movement among the service entities needed for the cooperative processing of the service entities is carried out by moving the service agent to the service entities assigned by the utilization orders according to the utilization orders of the service agent by using these functions.

[0029]

[Functions] The present invention can give the cooperative processing among application programs in a format independent of application program bodies being operation instruction (service agent) by taking the cooperative processing method among application programs as the above constitution. The cooperative interfaces for cooperative processing are added to the application program bodies, and the cooperative interface can easily change cooperative processing procedures among application pro-

grams through the service agent by executing utilization procedures held by the service agent.

[0030] The routine managers as a movement control means operating on each computer constituting the dispersion processing system established a communication path for sending/receiving the service agent and manage a communication path for sending/receiving the service agent between the names of service entities operating on the computers operated by the routine managers and the service entities. If the service entities are started, a communication path for sending/receiving the service agent among the routine managers on the computers operated by the service entities (application programs + cooperative interfaces) is established, names of the service entities are sent to the routine managers, the routine managers receiving the names manage a communication path to the names of delivered service entities and their service entities. The routine managers judge whether the service entities assigned by utilization orders of service agent received from the routine managers operating on other computers through the communication path are available on their own operating computer, sending the service agent to the pertinent service entities if they are available or moving the service entities to routine managers on other computers if they are unavailable.

[0031] If the service entities receive the service agent through the communication path, the cooperative interfaces acquire procedures and data used by the service entities from the utilization procedures held by the service agent, control the processing of application program bodies having the service entities by executing the acquired procedures, obtain processing results needed by the service agent, send the results to the service agent and store them in data items of the service agent. When it is assigned that programs corresponding to the procedures acquired from the service agent are on another computer, a cooperative interface copies the program on computers where the service entity operates and executes the procedures required by the service agent. After a processing result needed by the service agent is stored, the cooperative interface of the service entity renews the movement destination of the service agent and sends the service agent to a routine manager through the communication path. The routine manager judges whether the service entity assigned by the utilization order of the service agent sent from the service entity is available on its own operating computer, sends the service agent to the service entity if the entity is available or moves the service agent to a routine manager on another computer if the entity is unavailable.

[0032] Thus, the routine manager performs the cooperative processing method among application programs by moving a service entity assigned by utilization procedures constituting a dispersion processing system to a computer where the service entity operates according to utilization orders of a service agent, and the service entity acquires and executes a procedure held by a received service agent.

[0033] In the present invention, the cooperative processing among application programs can be easily changed through the service agent by giving a cooperative processing procedure among application programs in a format independent of an application program body (service agent), adding a cooperative interface for the cooperative processing to the application program body and

/7

executing the utilization procedures of the service agent held by this cooperative interface.

[0034] By using this cooperative processing method, the utilization of combining plural application programs made by operators, operations of the plural application programs made at a request of each operator and acquisition of the result are automatized before by describing utilization procedures combining various application programs necessary for advancing business and their operations as utilization procedures of applica-

tion programs used in a series operation procedures in the service agent, moving this service agent among various service entities operating on computers connected to a computer network and executing utilization procedures described in the service agent by the service entities received by the service agent, and the operating load of each operator can be reduced and the flow of operation can be smoothened by systematizing the utilization procedures of application programs necessary for a series of operations.

[0035] A program for preparing a production schedule, a program for simulating operations of a production system, etc. are considered as application programs suitable in the present invention. When production processes in a plant composed of plural production equipment controlled by computers are managed, respectively, there were such problems that "How many products (a lot) are in the plant?" and then "What equipment and what processing operations are performed?" must be accurately instructed for some products, technicians must make such instructions before by judging production progress conditions changing from time to time, thus it is complicated and has bad efficiency. The present invention enables to autonomously (independent of hands) make the judgments and instructions as described above while exchanging information among equipment providing services

and lots receiving it, thus this is suitable for the autonomy of the production system as described above. The present invention is also applicable to various services (on-line shopping, reservation of tickets, etc.) through a computer network.

[0036]

[Actual Examples] An actual example of the present invention will be described in detail by drawings below. In the present invention, a cooperative processing method among application programs in which processing is made progress will be described by cooperating application programs through plural service entities while moving a service agent among plural service entities composed of application program bodies and cooperative interfaces.

[0037] Fig. 1 is a diagram showing the constitution of a dispersion processing system operated by the cooperative processing method among application programs of this actual example. In this actual example, when some operations are performed by users in the above dispersion processing system, they must be processed by cooperating application programs AP1 → AP2 → AP3 or AP4 in this order. In the processing of each application program, AP1 is processed with data stored in item 1, item 2 of data items of a service agent (operation instruction: described later by Fig. 3) and the processing result is stored in item 6;

AP2 is processed with data stored in item 3, item 4 and the processing result is stored in item 7; AP3 is processed with data stored in item 4, item 6 and the processing result is stored in item 8; and AP4 is processed with data stored in item 5 and the processing result is stored in item 9.

[0038] This dispersion processing system comprises computers 1, a network 0 for performing data among the computers, application program bodies 3 operating on the computers, cooperative interfaces 30 for performing processing in cooperation with other application programs, service entities 4 being processing units as basis of cooperative processing among application programs for giving cooperative interfaces 30 to the application program bodies 3, a service agent 20 moving among the application programs to realize the cooperative processing, and routine managers 10 establishing and managing a communication path for moving the service agent 20 and a communication path among service entities and moving the service agent 20 among the service entities 4.

[0039] In the description of this actual example, the names of computers 1 are C1, C2, C3, respectively, the names of application program bodies 3 are AP1, AP2, AP3, AP4, respectively, and the names of service entities 4 having the application program bodies AP1 ~ AP4 are SE1, SE2, SE3, SE4, respectively.

When computers C1 ~ C3 must be differentiated, symbols 1a, 1b, 1c are used; when application programs AP1 ~ AP4 must be differentiated, symbols 3a, 3b, 3c, 3d are used; when service entities SE1 ~ SE4 must be differentiated, symbols 4a, 4b, 4c, 4d are used.

[0040] A constitutional example of computer 1 is shown in Fig. 2. The computer 1 comprises an input unit 13 composed of a keyboard 11 and a mouse 12, etc., a processing unit 16 being a computer main body stored with a CPU 14 and a memory 15, a display unit 17 displaying data, an external storage unit 18 and a printer 19 printing data.

[0041] The input device 13 may also be a touch panel other

/8

than the above. A scanner for inputting image data into the input unit 13 and a mike inputting voice may also be added. The mouse of input unit 13 is a means for assigning a selected target from a menu including some selection legs displayed on the display unit 17 or may be another unit having such a assigning means, e.g., an optical pen or a touch panel. A speaker outputting voice may also be added as a display unit 17.

[0042] The network 0 is a means sending and receiving data among the plural computers 1, and it may also be any network of LAN (Local Area Network) being a network in a specific place,

WAN (Wide Area Network) being a network among bases, an internet being a network connecting networks with each other.

[0043] One routine manager 10 shown in Fig. 1 exists in each computer and is started at the start-up of computer. For example, a method using a UNIX modem, etc. is given as a method for starting the routine manager 10 at the start-up of computer.

[0044] In the cooperative processing method among application programs based on this actual example, cooperative processing among application programs is carried out by moving the service agent 20 among the computers 1 connected to the network 0 through the routine managers 10, executing and controlling the processing of application program bodies 3 therein through the cooperative interfaces 30 of service entities 4 operating on these computers 1. Thus, a series operations are automatized and supported by performing the control and management of execution of the application program bodies 3 through the plural service entities 4 using the service agent 20 and cooperatively processing among the application programs used in a series of operations.

[0045] The cooperative processing method among application programs based on this actual example can be used by service entities 4 having two or more application program bodies 3 operating on one or more computers 1, but the cooperation of

three computers and four service entities 4 operating on the respective computers is described as an example in Fig. 1. Thus, plural service entities 4 can be included on one computer 1. The number of computers 1 and the number of service entities 4 having application program bodies 3 are not restricted to the illustrated constitution.

[0046] The structure of a typical service agent 20 used in the cooperative processing method among application programs based on this actual example is shown in Fig. 3. Different plural service agents 20 can flow in a system simultaneously. The service agent 20 is prepared by individual users on any computer. In addition, the prepared service agent 20 sometimes also generates a new service agent as necessary.

[0047] As shown in Fig. 3, the service agent 20 comprises an identifier 201 for differentiating individual service agents 20, a movement destination list summary 202 for describing a summary of movement destinations of service agent, a movement list summary 203 for managing paths through which the service agent moves, a movement destination name 204 for showing names of service entities 4 held by application program bodies 3 needed by the service agent 20 at the current point of time and names of computers where the service entities 4 operate, and an all data item 21 for listing all data items 211 ~ 219 needed by

the service entities 4 performing the cooperative processing, a data input procedure 22 for describing an input method corresponding to the data items 211 ~ 219 in the all data item 21, a data output procedure 23 for describing output methods corresponding to the data items 211 ~ 219 in the all data item 21 and service entities 4, an input data processing procedure 24 which is methods for processing data inputted by the data input procedure 22, a movement procedure 25 which is methods for determining a movement destination of this service agent 20 and sending the service agent 20 to the determined moving destination, a control procedure 26 which is a procedure for controlling the execution of application program bodies 3 having the service entities 4, and a generation procedure 27 which is a procedure for generating a new service agent.

[0048] In the structure of service agent 20, the identifier 201, movement destination list summary 202, movement list 203 and movement destination name 204 are items prepared certainly in the preparation of service agent 20. In addition, the items and procedures of all data item 21, data input procedure 22, data output procedure 23, input data processing procedure 24, movement procedure 25, control procedure 26 and generation procedure 27 are different elements in each cooperative processing

method among service entities 4 carried out by the service agent 20, and all the elements do not always exist.

[0049] The all data item 21 comprises inputted data item

/9

270 being data items (211 ~ 215) holding some values, an uninputted data item 280 being data items (216 ~ 219) holding no values. When the inputted data item 270 and the uninputted data item 280 are received in some service entity 4 by the service agent 20, an output data item 271 which is a data item storing data to be transferred to the service entity 4 and an input data item 281 which is a data item storing execution result data of the service entity 4. The input and output mentioned here are viewed from the service agent 20, the delivery of data from the service agent 20 to the service entity 4 is taken as an output and its reverse is taken as an input.

[0050] For example, if data needed in execution of the application program body AP1 is stored in the item 1 (211), item 2 (212) and the execution result is stored in item 6 (216), the item 1 (211) and item 2 (212) are output data items (271) and the item 6 (216) is an input data item (281) in the service entity SE1. If data needed in execution of the application program body AP2 is stored in the item 3 (213), item 4 (214) and the execution result is stored in item 7 (217), the item 3 (213)

and item 4 (214) are output data items (271) and the item 7 (217) is an input data item (281) in the service entity SE2. Similarly, if data needed in execution of the application program body AP3 is stored in the item 4 (214), item 6 (216) and the execution result is stored in item 8 (218), the item 4 (214) and item 6 (216) are output data items (271) and the item 8 (218) is an input data item (281) in the service entity SE3; if data needed in execution of the application program body AP4 is stored in the item 5 (215) and the execution result is stored in item 9 (219), the item 5 (215) is an output data item (271) and the item 9 (219) is an input data item (281) in the service entity SE4. Thus, the output data items (271) and the input data items (281) are different for each service entity received by the service agent 20.

[0051] The data input procedure 22 has an input data selection procedure 221 which is a procedure for selecting input data items 281 corresponding to the application program bodies 3 held by the service entities 4 from the uninputted data items 280, an input data acquisition procedure 222 which is a procedure for acquiring values form a picture by a user, a data acquisition procedure 223 which is a procedure for acquiring output result data of application program bodies 3, an input data conversion procedure 224 which is a procedure for converting the output

result data of application program bodies 3 acquired by this data acquisition procedure 223 to a data format of input data items 281, and an input data storage procedure 225 which is a procedure for storing values acquired by the input data acquisition procedure 222 and values converted in the data format of input data items 281 to the input data item 281.

[0052] ⌈Conversion of data format⌋ mentioned here means the conversion of data format. As specific examples, when some data have attribute names ⌈number⌋, ⌈unit price⌋, application programs output ⌈10⌋ and ⌈30⌋ as values of ⌈number⌋ and ⌈20⌋ and ⌈40⌋ as values of ⌈unit price⌋ in an ⌈output format of execution results of application programs⌋ as shown in Table 1 and store the values in the input data item in an ⌈format of input data items of service agent⌋ as shown in Table 2, the input data conversion procedure performs a conversion of format from the ⌈output format of execution results of application programs⌋ to the ⌈format of input data items of service agent⌋ as below.

[0053]

[Table 1]

≅ ⌈Output format of execution results of application programs⌋

(((number unit price)

(10 20))

((number unit price)

(30 40)))

[0054]

[Table 2]

≅ ⌈Format of input data item of service agent⌋

((number unit price)

(10 20))

(30 40))

[0055] The data output procedure 23 has an input data selection procedure 231 which is a procedure for selecting output data items 281 corresponding to the application program bodies 3 held by the service entities 4 from the inputted data items 270, an output data conversion procedure 232 which is a pro-

/10

procedure for converting values output data items 281 selected by this output data selection procedure, a parameter setup procedure 233 which is a procedure for transferring values of output data items 281 converted by the output data conversion procedure 232, and an output data display procedure 234 which is a procedure for displaying the values of output data items 281 converted by the output data conversion procedure 232 on the picture.

[0056] The input data processing procedure 24 is a procedure for processing the data acquired by the input data acquisition procedure 222 and the data acquisition procedure 223. For example, processing for calculating the sum of plural data acquired by the input data acquisition procedure 222, etc. are described as this input data processing procedure 24.

[0057] The movement procedure 25 is a procedure for determining a movement destination of service agent 20 and sending the service agent 20 to the determined movement destination. A method for determining the next movement destination according to a movement order given in the preparation of service agent and a method for determining a movement destination of service agent in accordance with any combinations of data items in the inputted data item 270 and the uninputted data item 280 are given as methods for determining the movement destination of the movement procedure 25. Namely, if the next movement destination is determined according to a movement order given in the preparation of service agent, acquire head elements of the movement destination list summary 202 of service agent 20, store the elements in the movement destination name 204, renew values of the movement destination list summary 202, and send the renewed service agent 20 to the routine managers 10 through an input/output channel 403 described later. If the next movement desti-

nation is determined in accordance with any combinations of data items, as shown in Fig. 21, examine values of noticed data items, store the movement destination determined by the values in the movement destination name 204, and send the service agent 20 with renewed movement destination name to the routine managers 10 through the input/output channel 403.

[0058] The control procedure 26 is a procedure for controlling the execution of application program bodies 3 executed by the cooperative interfaces 30 of service entities 4 and has a control script which is a combination of start and end of application program bodies 3, command execution of application program bodies 3 and plural commands.

[0059] The generation procedure 27 is a procedure for generating a new service agent 20 and has a control script generating the service agent 20.

[0060] Fig. 4 ~ Fig. 6 show a connection form of a logic communication path constructed by the routine managers 10 operating on each computer. The routine managers 10 perform establishment and maintenance of the logic communication path among the plural computers and establishment and maintenance of the input/output channel (Fig. 6: 403) among the service entities 4. The logic communication path is constructed from a combination of channels 40 which are logic communication paths connecting two

mutually connected routine managers 10. Fig. 4 shows that RM1 and RM2, RM2 and RM3 are connected among three routine managers 10 operating on computers C1, C2, C3 to construct an RM1-RM2-RM3 logic communication path. Fig. 5 is a diagram shows a state that a new routine manager 10 is connected to the routine manager 10 RM2 constructing the logic communication path shown in Fig. 4 to change the construction of logic communication path. Fig. 6 shows a connected state that the routine manager 10 RM3 and the service entities 4 SE3, SE4 in some computer are connected by the input/output channels 403 and a connected state that the cooperative interfaces 30 and the application program bodies 3 constructing the service entities 4 are connected by an application interface 38. The input/output channels 403 are communication paths connecting the routine managers 10 and the service entities 4 and are actually connected between the cooperative interfaces 30 in the service entities 4 and the routine manager 10. The application interface 38 is an interface connecting the input and output of the cooperative interfaces 30 generated in the service entities 4 and the application program bodies 3.

[0061] The routine managers 10 have channel ports 401 as sockets for connecting the channels 40. The channels 40 have channel numbers for identifying individual channel ports 401. The channel numbers are given in the generation of channel ports

401 by the routine managers 10. Namely, the channels 40 are communication paths connecting the channel ports 401 of two routine managers 10.

[0062] If the channel ports 401 are used by generating a channel 40, the routine managers 10 generates one new channel port 402. Therefore, the routine managers 10 always hold a vacant channel port 402 not connected with others. When a connection request is made from another routine manager 10, this vacant channel port 402 is used to generate a channel 40 among the routine managers 10.

/11

[0063] When the vacant channel port 402 generated by the routine managers 10 connects a service entity 4 and a routine manager 10, it is similarly used as in the case of connecting routine managers 10 with each other. Namely, if the input/output channels 403 are generated between the routine manager 10 and the service entities 4, a new vacant channel port is generated in the cooperative interfaces 30 of service entities 4 when the service entities 4 are started, a request for connection from the service entities 4 to the routine manager 10 is made, the routine manager 10 prepares the input/output channels 403 between the vacant channel port 402 and the vacant channel ports of cooperative interfaces 30 of service entities 4, and only the

routine manager 10 generates a new vacant channel port 402. If the channels 40 are thus generated, the routine managers 10 manages the channel ports so that one new vacant channel port 402 is generated and one vacant channel port 402 always remains.

[0064] The service entities 4 generate the vacant channel ports 402 for generating the input/output channels 403 between the service entities 4 and the routine managers 10 at the start, in addition to the generation of input/output channel 403 between the service entities 4 and the routine managers 10, the service entities 4 hold the cooperative interfaces 30 and the application interfaces 38 for connecting the cooperative interfaces 30 and the input/output of application program bodies 3, send the input/output between the cooperative interfaces 30 and the application program bodies 3, i.e., send commands from the cooperative interfaces 30 to the application program bodies 3 by this application interfaces 38, and receive execution result data of the application program bodies 3 in the cooperative interfaces 30.

[0065] The channel ports 401 have a buffering function during reception and hold the data sent through the channels 40 and the input/output channels 403 until a reading request exists.

[0066] Fig. 7 shows a structure of programs in the processing unit 16 for realizing the routine manager 10. The routine

manager 10 is composed of a management table preparation program 41 placed on a memory 15 of processing unit 16, a connection request program 42, a connection management program 43, a movement management program 44, a connection waiting program 45, a connection destination management table 46, a service entity management table 47, and a connection management table 48.

[0067] If the routine manager 10 placed on the memory 15 of processing unit 16 is started, first, the management table preparation program 41 reads a name summary of connection destination computers stored in the external storage unit 18 and prepares the connection destination movement management table 46 on the memory 15 of processing unit 16. Next, it reads a name summary of operating service entities by the computers operated by the routine manager 10 and prepares the service entity management table 47 on the memory 15 of processing unit 16.

[0068] The connection request program 42 acquires connection destination computer names in order from the head of a summary of computer names stored in the connection destination movement management table 46 prepared on the memory 15 of processing unit 16 and requests a connection to the connection destination computers. If the connection cannot be made, it acquires the next connection destination computer name and successively makes connection requests until the connection can be made. If

the connection cannot be made even a connection request is made, the connection request program 42 outputs an error message and ends the processing.

[0069] If the connection management program 43 receives such a report that the connection with the computers (connection destination computers) connected by the connection request program 42 is completed, the connection management program 43 sends the names of computers making the connection requests (connection source computers) to connection destination computers and waits the names of connection destination computers transmitted from the connection destination computers. If the names of connection destination computers sent from the connection destination computers are received, store pairs of the names of connection destination computers and channel port numbers of the channels 40 connected to the connection destination computers in the connection management table 48. If the control is transferred by receiving the channel port numbers, the connection waiting program 45 described later waits for the names of connection destination computers or names of service entities 4 sent from the channel port numbers. If the names of connection destination computers or names of service entities 4 are received, store pairs of the numbers and the channel port numbers of a logic communication path therebetween in the connection management

table 48. If the delivered names are names of connection source computers, transmit the connection destination computer names to the connection source computers.

[0070] If the connection management program 43 stores the pairs of the numbers of connection source computers and the channel port numbers in the connection management table 48 or connection requests are not made at an interval of given time, the movement management program 44 is started. The movement management program 44 successively acquires channel port numbers from the head of connection management table 48, reads the ser-

/12

vice agent 20 from the channel 40 connected to the channel port numbers 401 and checks whether the names of service entities 4 assigned by the movement destination name 204 of read-out service agent 20 are stored in the connection management table 48. When the names are stored, send the service agent 20 to the service entities 4. If the names are stored, send the service agent 20 to the service entities 4. If the names are not stored, retrieve other connected routine managers 10 from the connection management table 48 and send the service agent 20 to resultant routine manager 10. If no service agent 20 is read from the channel 40, read the service agent 20 similarly from the channel 40 connected to the next channel port number 401. Repeat it

until the channel port numbers 401 stored in the connection management table 48 disappear. If all the channel port numbers stored in the connection management table 48 are successively read out, transfer the control to the next connection waiting program 45.

[0071] The connection waiting program 45 waits for a connection request from other routine managers 10 or service entities 4 (wrong number "30", translator) for some given time. If a connection request exists at an interval of some given time, connect the channel 40 among the routine managers 10 of the connection request source or connect the input/output channels 403 among the service entities 4 of the connection request source, generate a new vacant channel port 402, and send the channel port number of channel port 401 of newly connected channel 40 to the connection management program 43. If no connection request is made in an interval of some given time, transfer the control to the movement management program 44.

[0072] Fig. 8(a) shows the structure of connection destination management table 46, Fig. 8(b) shows the structure of service agent management table 47, and Fig. 8(c) shows the structure of connection management table 48.

[0073] The connection destination management table 46 has a connection destination computer name field 460 keeping a name

list of connection destination computers. The connection destination computer name field 460 has computer names 461 which are names of computers making a connection request at the start of routine managers 10 as values. Information of own computer is certainly not included in this connection destination management table 46.

[0074] The service entity management table 47 has a service entity name field 470 keeping a name summary of service entities 4 operating on computers operated by the routine managers 10. The service entity name field 470 has names 471 of service entities 4 operating on computers operated by the routine managers 10 as values.

[0075] The connection management table 48 has a connection destination name field 480 keeping the names of computers operated by other routine managers 10 connected by the channel 40 and the names of service entities 4 connected by the input/output channels 403, a channel port number field 481 keeping channel port numbers of channel ports 401 connected by the channel 40 and the input/output channels 403 used for connection with these connection destinations, and a connection destination kind field 484 for differentiating whether the connection destinations are routine managers 10 or service entities 4. The connection destination name field 480 has connection destination

names 482 connected to the routine managers 10 by the channel 40 and the input/output channels 403 as values, and the channel port number field 481 hold channel port numbers 483 connected by the channel 40 and the input/output channels 403 corresponding to the connection destination names 482. A connection destination kind field 484 takes either values of routine managers 10 showing that the connection destinations of channels are routine managers 10 and service entities 4 showing that the connection destinations of channels are service entities. These two values are put together and called as connection destination kind names 485.

[0076] Fig. 9 shows a structure of programs in the processing unit for realizing the cooperative interface 30. The cooperative interface 30 is composed of programs of a data preparation program 31, an output data program 32, an execution control program 33, a data acquisition program 34, a data storage program 35, a movement destination operating program 36, a function control program 37 and a function table 371 placed on the memory 15 of processing unit 16. The programs need not to be kept in the service agent 20 itself by providing the function table 371, thus the load during the transmission of service agent 20 among the computers can be reduced.

[0077] If the service agent 20 is received through the input/output channels 403, the cooperative interface 30 transfers the received service agent 20 to the data preparation program 31.

[0078] The data preparation program 31 selects the output data item 271 which is a data item storing data transferred from the inputted data item 270 to the application program body 3 by the output data selection procedure 231 in the received service agent 20.

[0079] If the output data item 271 is determined, prepare data transferred to the application program body 3 or data out-

/13

putted to the picture from data stored in the output data item 271 by the output data conversion procedure 232, attach a tag of either data and transfer the prepared data to the data output program 32.

[0080] If the data is transferred to the data output program 32, judge whether the data output program 32 judges the data are outputted from the tag of received data to the picture or sent to the application program body 3, and it outputs the data to a picture by the output data display procedure 234 of the service agent 20 in case of outputting the data to the screen or transfers the data to the execution control program 3

in case of transferring the data to the application program bodies 3. When the output data display procedure 234 described in the service agent 20 is not a program but simply a function name, the data output program 32 transfers the function name to the function control program 37.

[0081] If the function control program 37 receives the function name, it retrieves a function pertinent to the function name from the function table 371 and executes the retrieved program. If the execution control program 33 receives the data from the data output program 32, it executes the control procedure 26 of service agent 20 and controls the execution of application program body 3. When the function control program 37 executes the application program body 3, it sends the data to the application program body 3 by the parameter setup procedure 233 of service agent 20 in case the data must be transferred to the application program body 3. If the execution of application program body 3 is controlled and the execution of control procedure is ended, acquire the execution result data of application program body 3 by the data acquisition procedure 223 of service agent 20 and transfer the acquired data to the data acquisition program 34. If the control procedure 26 is a function name or processing of only a function name is included in the control

procedure 26, transfer the function name to the function control program 37 and execute a pertinent function program.

[0082] If the data acquisition program 34 receives the execution result data, select the input data item 281 which is a data item storing the execution result data of application program body 3 from the all data item 21 of service agent 20 by the input data selection procedure 221 of service agent 20. When a picture is outputted, acquire only necessary data in the data inputted to the picture by the input data acquisition procedure 222 of service agent 20. Transfer these acquired data (execution result data of application program body 3, input data from the picture) and the input data item 281 to the data storage program 35. If the input data acquisition procedure 222 described in the service agent 20 is not a program but only a simple function name, transfer the function name to the function control program 37 by the data acquisition program 34 and executes the program of function of the pertinent function name.

[0083] If the data storage program 35 receives the data from the data acquisition program 34 and the input data item 281, in case the input data processing procedure 24 exists, process data stored in the received data and the all data item 21 by the input data processing procedure 24 with reference to the region of input data processing procedure 24 of the service

agent 20, convert the data after processing to a data format of received input data item 281 by the input data conversion procedure 224 and store the converted data to respective data items of the input data item 281. In case the input data processing procedure 24 does not exist, convert the data after processing to a data format of received input data item 281 by the input data conversion procedure 224 and store the converted data to respective data items of the input data item 281.

[0084] The movement destination operating program 36 stores the data in the input data item 281 calls it therefrom, acquires head elements of the movement destination list summary 202, stores them in the movement destination name 204, and sends the service agent 20 with a movement destination name 204 renewed in this manner to the routine managers 10 through the input/output channels 403.

[0085] Fig. 10(a) shows a structure of function table 371 used in the function control program 37. The function table 371 comprises a function name field 372 and a processing field 373 which is a processing program body of function. Either a program 373a describing processing contents of function or a storage position of program 373b describing processing contents of the function are described in the processing field 373.

[0086] If the processing field 373 of pertinent function name is the program 373a, the function control program 37 executes the program. If it is the storage position of program 373b, read (copy) the program 374 at the storage position on the memory 15 of processing unit 16 through the network 0 and then execute the program. Programs on other computers can be shared by keeping the storage position of program rather than the program itself in such a function table. It has advantages of reducing the memory capacity of the function table and changing

/14

the program in only one place in case of program change.

[0087] The storage position of program 373b is assigned by the description format of Fig. 10(b). The storage positions are assigned in the form of a computer name 375a stored by the program and a storage pass 375c indicating the storage position of program on a computer assigned by the computer name 375a, and assigned in the form of computer name 375a, breakoff symbol 375b and storage pass 375c by using the breakoff symbol 375b for differentiating the computer name 375a and storage pass 375c.

[0088] Fig. 11(a) shows the description format of the movement destination list 91 stored in the movement destination list summary 202 of the service agent 20, and Fig. 11(b) shows its specific description example.

[0089] The movement destination list 91 connects a movement destination computer name 912 which is names of movement destination computers of the service agent 20 and a movement destination 911 which is a group of service entity names 913 being names of service entities 4 utilized by the movement destination computers with arrows (→) according to a movement order of the service agent 20. When the movement destination computer name are unknown, Unknown is assigned to the movement destination computer name 912.

[0090] In the example of Fig. 11(b), it is shown that the service agent 20 performs processing by an application program body 3 AP1 having a service entity 4 named as SE1 of a computer named as C1, then performs processing by an application program body 3 AP2 having a service entity 4 named as SE2 of a computer named as C2, next performs processing by an application program body 3 AP3 having a service entity 4 named as SE3 of a computer named as C3 and finally performs processing by an application program body 3 AP4 having a service entity 4 named as SE4 although the name of a computer operating the service entity 4 SE4 is unclear.

[0091] Fig. 12(a) shows the description format 101 of the movement destination list stored in the movement list summary

203 of the service agent 20, and Fig. 12(b) shows its specific description example.

[0092] The movement list summary 203 seeks "Which computer on a network does a service utilized by the service agent 20 operates on?" and is used in movement. More specifically, the movement list summary 203 is used when the movement destination computer name 912 of the movement destination list shown in Fig. 11 is Unknown. If the movement destination computer name 912 is Unknown, the routine manager 10 of each computer checks "Whether a service entity 4 having a name assigned by the movement destination name 204 of service agent 20 exists?", if it does not exist, stores the computer name in the movement destination computer name 102 and sends the service agent 20 to the routine manager 10 on another computer.

[0093] When the movement destination computer name 912 is Unknown, the movement list 101 connects the name of computer, where service entity 4 assigned by the movement destination names 204 of service agent 20 does not exist, with arrows (→) according to a movement order of the service agent 20. The movement list 101 successively supplements a computer name every time the service agent 20 moves and cleared at the time of finding a computer where the service entity 4 assigned by the movement destination name 204 of service agent 20 operates.

Thus, computers once sought are stored in this list 101, and many times of retrievals of the same computer can be prevented. The movement list summary 203 is useful to avoid seeking services utilized by the service agent 20 and visiting the same computer many times to perform an effective search of services because there is a possibility of having such an event if this list does not exist.

[0094] The example of Fig. 12(b) shows that the service entity 4 assigned by the movement destination name 204 of service agent 20 does not exist in the computer named as C1 and then also does not exist in the computer named as C2.

[0095] Fig. 13 ~ Fig. 16 show flows of movement processing procedures of service agent 20 performed by the routine managers 10 to realize the cooperative processing method among application programs using the service agent 20.

[0096] The service agent 20 is sent to the routine managers 10 of computer prepared thereby. At this time, the value of movement destination list summary 202 of service agent 20 is a movement order shown in Fig. 11(b). The value of movement destination name 204 of the service agent 20 at this time is taken as the head element SE1 (911a) of movement destination list 91 which is the value of movement destination list summary 202.

[0097] If the service agent 20 is received (1100), first, the routine manager 10 checks whether the movement destination computer name of movement destination name 204 of the service agent 20 is same as the name of computer where the routine manager 10 operates (1101).

[0098] If the movement destination computer name of movement destination name 204 of service agent 20 and the name of computer where the routine manager 10 operates are different, check whether the connection destination in which the connection destination name 482 of connection destination name field 480 is same as the movement destination computer name is registered from connection destinations in which the movement destination kind name 485 of movement destination kind field 484 of the connection management table 48 is a routine manager (1102). If the movement destination computer name is registered, acquire a channel port number 483 stored in the channel port number field 481 of the connection destination name 482 pertinent to the movement destination computer name (1103), and send the service agent 20 to the routine manager 10 of movement destination computer through the channel 40 connected to the channel port number 483 (1104). If the movement destination computer name is not registered, acquire a channel port number 483 stored in the channel port number field 481 of the first registered connection desti-

nation name 482 from connection destinations in which the movement destination kind name 485 of movement destination kind field 484 of the connection management table 48 is a routine manager (1105), and send the service agent 20 to the routine manager 10 of registration destination computer through the channel 40 connected to the channel port number 483 (1106).

[0099] When the movement destination computer name 912 of movement destination name 204 of the service agent 20 and the name of computer in which the routine manager 10 operates are the same, check whether the service entity in which the connection destination name 482 of connection destination name field 480 is same as the service entity name of movement destination name 204 is registered from connection destinations in which the movement destination kind name 485 of movement destination kind field 484 of the connection management table 48 is the service entity (1107). If the service entity name is registered, acquire a channel port number 483 stored in the channel port number field 481 of the registered service entity (1108), and send the service agent 20 to a cooperative interface 30 of service entity 4 through the input/output channels 403 connected to the channel port number 483 (1109).

[0100] If the service entity name is not registered in the connection management table 48, check whether the service entity

name is registered in the service entity management table 47 (1110). If the service entity name is registered in the service entity management table 47, start the service entity 4 (1111). The cooperative interface 30 of the started service entity 4 makes a connection request to the routine manager 10. The routine manager 10 waits for the connection request from the cooperative interface 30 of service entity 4 (1112), if the connection request is received, generate input/output channels 403 between the routine manager 10 and the cooperative interface 30 (1113), register the name of started service entity and the channel port numbers of the input/output channels 403 in the connection management table 48 (1114), and sent the service agent 20 to the cooperative interface 30 of service entity 4 through the generated input/output channels 403 (1109). If the service entity name is not registered in the service entity management table 47, change the movement destination computer name 912 of movement destination name 204 of the service agent 20 to Unknown (1115a), acquire the channel port number 483 stored in the channel port number field 481 of the first registered connection destination name 482 (1105), and send the service agent 20 to the routine manager 10 of registration destination computer connected to a channel through the channel 40 connected to the channel port number 483 (1106).

[0101] If the movement destination computer name 912 of movement destination name 204 of the service agent 20 is Unknown, shift the flow to Fig. 14, check whether the connection destination in which the connection destination name 482 of connection destination name field 480 is same as the service entity name of movement destination name 204 of the service agent 20 is registered from connection destinations in which the movement destination kind name 485 of movement destination kind field 484 of the connection management table 48 is the service entity (1115b). If the service entity name is registered, acquire the channel port number 483 stored in the channel port number field 481 of the registered service entity (1116), and send the service agent 20 to the cooperative interface 30 of service entity 4 through the input/output channels 403 connected to the acquired channel port number 483 (1117). If the service entity name is not registered in the connection management table 48, check whether the service entity name is registered in the service entity management table 47 (1118). If the service entity name is registered in the service entity management table 47, start the service entity 4 (1119). The cooperative interface 30

/16

of the started service entity 4 makes a connection request to the routine manager 10. The routine manager 10 waits for the

connection request from the cooperative interface 30 of service entity 4 (1120), if the connection request is received, generates input/output channels 403 between the routine manager 10 and the cooperative interface 30 (1121), register the name of started service entity and the channel port number of the input/output channels 403 in the connection management table 48 (1122), and sent the service agent 20 to the cooperative interface 30 of service entity 4 through the generated input/output channels 403 (1117).

[0102] If the service entity name is not registered in either the connection management table 48 or the service entity management table 47, shift the flow to Fig. 15, check whether a connection destination in which the movement destination kind name 485 of movement destination kind field 484 of the connection management table 48 is a routine manager exists (1123). If the connection destination is registered, fetch a first registered connection destination name 482 therein (1124), and check whether the connection destination name 482 is in the movement computer name 102 of movement list 101 stored in the movement list summary 203 (1125). If the connection destination name 482 is not in the movement list 101, supplement the name of current computer to the movement computer name 102 of movement list 101 (1126), store the supplemented movement list 101 in the movement

list summary 203 of service agent 20 (1127), fetch the channel port number 483 stored in the channel port number field 481 of the connection destination name 482 (1128), and send the service agent 20 to the routine manager 10 of a computer connected to the channel through the channel 40 connected to the channel port number (1129). If the connection destination name 482 is in the movement list 101, check whether the next connection destination in which the movement destination kind name 485 of movement destination kind field 484 of the connection management table 48 is a routine manager (1123). If the next connection destination is registered in the connection management table 48, fetch the connection destination name 482 (1124) and check whether the connection destination name 482 is in the movement computer names 102 of the movement list 101 stored in the movement list summary 203 of service agent 20 (1125). This is successively repeated until the connection destination in which the movement destination kind name 485 of movement destination kind field 484 of the connection management table 48 is a routine manager 10 disappears or a connection destination name 482 not in the movement list 101 is found. As a result, if the connection destination in which the movement destination kind name 485 of movement destination kind field 484 of the connection management

table 48 is a routine manager 10 disappears, end the movement of service agent 20 as an error (1130).

[0103] Shift the flow to Fig. 16, the cooperative interface 30 of service entity 4 receiving the service agent 20 executes a data output procedure described in the received service agent 20 and acquires data sent from the service agent 20 to the service entity 4 (1131), executes a control procedure 26 and controls the execution of application program body 3 having the service entity 4 (1132), executes a data input procedure 22 of service agent 20 and stores the execution result data in the service agent 20 (1133), executes a movement procedure 25 of service agent 20 and write a value of movement destination name 204 in the next movement destination (1134), and sends the service agent 20 written with the value of movement destination name 204 from the cooperative interface 30 to the routine manager 10 through the input/output channels 403 (1135). Return the flow to Fig. 13, if the routine manager 10 receives the service agent 20 sent from the cooperative interface 30 through the input/output channels 403 (1100), similarly as the case of receiving the service agent 20 through the channel 40, check a movement destination computer name 912 of movement destination name 204 (1101) and send the service agent 20 to the next movement destination through the channel 40 (1104). The service agent 20 is moved

among the service entities 4 by repeating the movement processing of this service agent 20 among routine managers 10 until it becomes the last element of movement destination list 91, and the execution of application program bodies 3 is controlled by processing the moved service agent 20 with the cooperative interfaces 30 of the service entities 4.

[0104] Fig. 17 ~ Fig. 20 show processing flows for controlling the execution of application program bodies 3 while utilizing data and procedures held by the received service agent 20 in the cooperative interfaces 30 of the service entities 4.

[0105] First, in Fig. 17, if a cooperative interface 30 receives the service agent 20 from a routine manager 10 through the input/output channels 403 (1200), execute an output data selection procedure 231 of the received service agent 20 and determine an output data item 271 which is a data item storing data transferred from the inputted data item 270 to application program bodies 3 (1201).

/17

[0106] If the output data item 271 is determined, check whether the output data conversion procedure 232 of the service agent 20 is a program or a function name (1202). If the output data conversion procedure 232 is a program, check whether the output data conversion procedure 232 is a data output function

or a picture output function to the application program bodies 3 (1203); if it is a data output function, execute the function and generate data transferred from data stored in the output data item 271 to the application program bodies 3 (1204). If it is a picture output function, execute the function and prepare data outputted from data stored in the output data item 271 to a picture (1205). Attach a tag indicating which data it is for the generated data (1206). If the output data conversion procedure 232 is a function name, acquire a program of the function name from a function table through the function control program 37 (1207) and check whether the obtained function is an data output function or a picture output function to the application program bodies 3.

[0107] Judge whether the tag-attached data is outputted from the tag to a picture or transferred to the application program bodies 3 (1208), in the case of data outputted to a picture, check whether the output data procedure 234 of service agent 20 is a program or a function name (1209). If the output data procedure 234 is a program, execute the output data procedure 234 and output a picture output data (1210). If the output data procedure 234 is a function name, acquire a program of the function name from the function table through the function control program 37 (1211), execute the acquired function and output

the picture output data to a picture (1210). In the case of data transferred to the application program bodies 3, delete the tag of data (1212).

[0108] Next, shift the flow to Fig. 18 and check whether the control procedure 26 of service agent 20 is a program or a function name (1213). If the control procedure 26 is a program, control the execution of application program body 3 by executing the control procedure 26 (1214). If the control procedure 26 is a function name, acquire a program of the function name from the function table through the function control program 37 (1215) and control the execution of application program bodies 3 by executing the acquired function (1214).

[0109] If the execution control of the application program bodies 3 is ended by the control procedure 26, check whether the data acquisition procedure 223 of service agent 20 is a program or a function name (1216). If the data acquisition procedure 223 is a program, execute the data acquisition procedure 223 and acquire execution result data of the application program bodies 3 (1217). If the data acquisition procedure 223 is a function name, acquire a program of the function name from the function table through the function control program 37 (1218), execute the acquired function and acquire execution result data of the application program bodies 3 (1217).

[0110] If a picture is outputted, shift the flow to Fig. 20, and check whether the control procedure 26 of service agent 20 is a program or a function name (1219). If the control procedure 26 is a program, control the input/output from the picture by executing the control procedure 26 (1220). If the control procedure 26 is a function name, acquire a program of the function name from the function table through the function control program 37 (1221) and control the input/output from the picture by executing the acquired function (1220).

[0111] If a picture is outputted, check whether the data acquisition procedure 223 of service agent 20 is a program or a function name (1222). If the data acquisition procedure 223 is a program, execute the data acquisition procedure 223 and acquire input data from the picture (1223). Namely, incorporate instructions or data inputted by a user through a picture. If the data acquisition procedure 223 is a function name, acquire a program of the function name from the function table through the function control program 37 (1224), execute the acquired function and acquire input data from the picture (1223).

[0112] Next, return the flow to Fig. 18, and check whether the input data selection procedure 221 of service agent 20 is a program or a function name (1225). If the input data selection procedure 221 is a program, execute the input data selection

procedure 221 and select an input data item 281 which is a data item storing processing result data of application program bodies 3 from the all data item 21 of service agent 20 (1226). If the data acquisition procedure 223 is a function name, acquire a program of the function name from the function table through the function control program 37 (1227), execute the acquired function and select an input data item 281 which is a data item storing processing result data of application program body 3 from the total data item 21 of service agent 20 (1226).

[0113] Next, shift the flow to Fig. 19, and check whether the input data processing procedure 24 of service agent 20 is a program or a function name (1228). If the input data processing

/18

procedure 24 is a program, execute the input data processing procedure 24 and processing the received data and data stored in the all data item 21 (1229). If the data acquisition procedure 223 is a function name, acquire a program of the function name from the function table through the function control program 37 (1230), execute the acquired function and process the received data and data stored in the all data item 21 (1229).

[0114] Next, check the input data conversion procedure 224 is a program or a function name (1231). If the input data conversion procedure 224 is a program, execute the input data con-

version procedure 224, convert the received data to the data format of an input data item 281 pertinent thereto (1232) and store the converted data in respective data items of the input data item 281 (1234). If the data acquisition procedure 223 is a function name, acquire a program of the function name from the function table through the function control program 37 (1233), execute the acquired function and convert the received data to the data format of an input data item 281 pertinent thereto (1232) and store the converted data in the respective data items of the input data item 281 (1234).

[0115] If the data is stored in the input data item 281, check whether the movement procedure 25 of service agent 20 is a program or a function name (1235). If the movement procedure 25 is a program, execute the movement procedure 25, rewrite the value of movement destination name 204 in the next movement destination (1236) and send the service agent 20 with rewritten the value of movement destination name 204 from the cooperative interfaces 30 to the routine manager 10 through the input/output channels 403 (1237). If the data acquisition procedure 223 is a function name, acquire a program of the function name from the function table through the function control program 37 (1238), rewrite the value of movement destination name 204 in the next movement destination (1236) and send the service agent 20 with

rewritten value of movement destination name 204 from the cooperative interfaces 30 to the routine manager 10 through the input/output channels 403 (1237).

[0116] Next, an example of applying the present invention to a production line management system is described as its specific application example. In the production line management system, production and inspection machines, application programs for control of these machines and application programs necessary for production management must be cooperated in a determined order for each product.

[0117] Fig. 22 shows the system constitution of one application example of a production line management system based on the present invention. In the application example, the production line management system is composed of computers C2 (1342), C3 (1343) for managing production processes A, B, production machines MA1 (1332), MB1 (1333) connected to production processes A, B, application programs AP2 (1302), AP3 (1303) for controlling these production machines MA1 (1332), MB1 (1333), an application program AP1 (1301) for preparing a production instruction on the computer C1 and a network 0 connecting the computers, and the production instruction prepared by the application program AP1 on the computer C1 is described by cooperatively controlling the production machines connected to produc-

tion processes with a case of advancing the production as an example.

[0118] In the production line management system based on this actual example, these various application programs AP1 (1301), AP2 (1302), AP3 (1303) are realized as service entities SE1 (1321), SE2 (1322), SE3 (1323) (name of each service entity is SE1, SE2, SE3, respectively). The production instruction attached to a product is embodied as a service agent SA1 (1400). A user prepares the service entity SA1 (1400) embodying the production instruction by using the service entity SE1 (1321), first, the user transmits a production instruction to the application program AP2 (1302) through the service entity SE2 (1322) having the application program AP2 (1302) for controlling the production machine MA1 (1332) (wrong number "1333" in original document, translator) of production process A, next transmits the production instruction to the application program AP3 (1303) through the service entity SE3 (1323) having the application program AP3 (1303) for controlling the production machine MB1 (1333) of production process B, subsequently returns the flow to the service entity SE1 (1321) with a prepared service agent SA1 (1400).

[0119] Fig. 23 shows the structure of service agent SA1 (1400) which is one example of service agent expressing a pro-

duction instruction at a time that of preparing it on the computer C1 (1341) by using the service entity SE1 (1321). The service agent SA1 (1400) has an identifier ID1 (1401), movement destination list summaries (C2 SE2) (C3 SE3) (C1 SE1) (1402), a movement list summary nil (1403), a movement destination name (C2 SE2) (1404), in addition, a process A production speed item (1405) which is an item being a setup parameter of production machine MA1 and its value 10, a process A required time item (1406) which is an item of data acquired from the production

/19

machine MA1, a process B production speed item (1407) which is an item being a setup parameter of the production machine MB1 and its value 15 and a process B production speed item (1408) which is an item of data acquired from the production machine MB1 as all data item. The service agent SA1 (1400) also has an output data selection procedure (1421), an output data conversion procedure (1422) and a parameter setup procedure (1423) as data output procedures, and a data acquisition procedure (1411), an input data selection procedure (1412) and an input data storage procedure (1413) as data input procedures and further has a control procedure (1430).

[0120] The service agent SA1 (1400) generated by the service agent SA1 (1321) is sent to the computer C1 (1341) through

the channel 40. A routine manager RM1 (1311) checks the value (C2 SE2) (1404) of movement destination name of the service agent SA1 (1400), retrieves the channel 40 connected with the computer C2 (1342) assigned by the movement destination name and sends the service agent SA1 (1400) to the computer C2 (1342) through the resultant channel 40.

[0121] If the routine manager RM2 (1312) on the computer C2 (1342) receives a delivered service agent SA1 (1400), check the value of movement destination name (C2, SE2) (1404), retrieve the channel 40 connected with the service entity SE2 (1322) assigned by the movement destination name and send the service agent SA1 (1400) to the service entity SE2 (1322) through the resultant channel 40. If the service entity SE2 (1322) receives the service agent SA1 (1400) through the channel 40, the cooperative interface 30 of the service entity SE2 (1322) acquires an output data selection procedure 1421 held by the service agent SA1 (1400), execute the procedure 1421 and determine a process A production speed item 1405 which is a data item to be transferred to the application program AP2 (1302) held by the service entity SE2 (1322). Next, acquire an output data conversion procedure 1422 held by the service agent SA1 (1400), execute the procedure 1422, fetch a value 10 of the process A production speed item 1405 which is a selected item, attach a tag indica-

ting that the data is transferred to an application program, then acquire a parameter setup procedure 1423 held by the service agent SA1 (1400), execute the procedure 1423 and set up the value 10 assigned by the application program AP2 (1302). Then, processing of the application program AP2 (1302) is performed by acquiring the control procedure 1430 held by the service agent SA1 (1400) and executing the procedure 1430.

[0122] If the processing of application program AP2 (1302) is ended, acquire the data acquisition procedure 1411 held by the service agent SA1 (1400), execute the procedure 1411 and acquire execution result of the application program. Next, acquire the input data selection procedure 1412 held by the service agent SA1 (1400), execute the procedure 1412, select the process A required time 1406 which is a data item storing the execution result of the application program, then acquire the input data storage procedure 1413 held by the service agent SA1 (1400), execute the procedure 1413, and store the execution result of the application program AP2 (1302) in the process A required time item 1406.

[0123] Next, fetch a second element (C3, SE3) of the movement destination list summary 1402, store it in the movement destination name 1406 and send it to the routine manager RM2 (1312) through the channel 40.

[0124] If the routine manager RM2 (1312) receives the service agent SA1 (1400) sent from the service entity SE2 (1322) through the channel 40, check its movement destination name (C3, SE3) (1404), retrieve the channel 40 connected to a computer C3 (1343) assigned by the movement destination name (C3, SE3) (1404), and send the service agent SA1 (1400) through the resultant channel (40).

[0125] If the routine manager RM3 (1313) on the computer C3 receives the delivered service agent SA1 (1400), check its movement destination name (C3, SE3) (1404), retrieve the channel 40 connected to the service entity SE3 (1323) assigned by the movement destination name (C3, SE3) (1404), and send the service agent SA1 (1400) to the service entity SE3 (1323) through the resultant channel (40).

[0126] If the service entity SE3 (1323) receives the service agent SA1 (1400) through the channel 40, the cooperative interface 30 of the service entity SE3 (1323) acquires the output data selection procedure 1421 held by the service agent SA1 (1400), executes the procedure 1421 and determines a process B production speed item 1407 which is a data item to be transfer-

/20

red to the application program AP3 (1303) held by the service entity SE3 (1323). Next, acquire the output data conversion pro-

cedure 1422 held by the service agent SA1 (1400), execute the procedure 1422, fetch a value 15 of the process B production speed item 1407 which is a selected item, attach a tag indicating that the data is transferred to the application program, then acquire the parameter setup procedure 1423 held by the service agent SA1 (1400), execute the procedure 1423 and set up the value 15 assigned by the application program AP3 (1303). Then, processing of the application program AP3 (1303) is performed by acquiring a control procedure 1430 held by the service agent SA1 (1400) and executing the procedure 1430.

[0127] If the processing of application program AP3 (1303) ends, acquire the data acquisition procedure 1411 held by the service agent SA1 (1400), execute the procedure 1411 and acquire execution result of the application program. Next, acquire the input data selection procedure 1412, execute the procedure 1412, select the process B required time item 1408 which is a data item storing execution result of the application program, then acquire the input data storage procedure 1413 held by the service agent SA1 (1400), execute the procedure 1413, and store execution result of the application program AP3 (1303) in the process B required time item 1408.

[0128] Next, fetch the (C1 SE1) which is the third element of the movement destination list summary 1402, store it in the

movement destination name 1404 and send it to the routine manager RM3 (1313) on the computer C3 (1343) through the channel 40.

[0129] If the routine manager RM3 (1313) receives the service agent SA1 (1400) sent from the service entity SE3 (1323) through the channel 40, check the movement destination name (C1 SE1) (1404) and retrieve the channel 40 connected to the computer C1 (1341) assigned by the movement destination name (C1 SE1) (1404), but the computer name C1 (1341) is not in the computer connected to the routine manager RM3 (1313) on the computer C3 (1343), therefore send the service agent SA1 (1400) to the routine manager RM2 (1312) of connected computer C2 (1342).

[0130] If the routine manager RM2 (1312) on the computer C2 (1342) receives the delivered service agent SA1 (1400), check the value of movement destination name (C1 SE1) (1404), the computer name C1 assigned by the movement destination name 1404 is different from the computer name C2, therefore store the computer name C2 in the movement list 1403 of the service agent SA1 (1400), take the value of movement list 1403 as ((C2)), retrieve the channel 40 connected to the computer name C1 assigned by the movement destination name (C1 SE1) (1404) of the service agent SA1 (1400) and send the service agent SA1 (1400) to the resultant channel 40.

[0131] If the routine manager RM1 (1311) receives the service agent SA1 (1400) sent from the routine manager RM1 (1311) through the channel 40, check the movement destination name (C1 SE1) (1404), check whether the movement destination name 1404 and the computer name C1 are the same, then retrieve the channel 40 connected to the service entity SE1 (1321) assigned by the movement destination name (C1 SE1) (1404), and send the service agent SA1 (1400) to the service entity SE1 (1321) through the resultant channel 40.

[0132] In this manner, the application programs AP2 (1302), AP3 (1303) for controlling the manufacturing machine MA1 (1332) connected to the computers C2 (1342), C3 (1343) managing the manufacturing processes A, B can be cooperatively controlled. Even if the manufacturing processes are different, manufacturing equipment for control and their order can be changed only by changing the summary of movement destinations assigned by the movement destination list summary, thus the cooperative method can be easily changed.

[0133] As described above, the present invention enables cooperative processing among flexible application programs by giving cooperative interfaces for the cooperative processing to application program bodies, describing cooperative processing procedures necessary for the cooperation and data necessary at

the time of cooperation, and storage data items and control procedures for execution of individual application program bodies in an operation instruction and moving the operation instruction among computers in order to flexibly process various application programs.

[0134] Thereby, the present invention enables to change the cooperative processing procedures among application programs by the operation instruction without stopping the execution of application programs as targets of the cooperative processing. Moreover, a user can utilize the application programs by assign-

/21

ing application programs to be utilized without being aware of "By which computer on a dispersion system are application programs to be cooperatively processed operated?".

[0135] The present invention also enables to automatize a series operations depending upon many operators and made by operating application programs by the operators.

[Brief Description of the Drawings]

[Fig. 1] Block diagram showing the constitution of actual example of a dispersion processing system performing cooperative processing among application programs based on the present invention.

[Fig. 2] Block diagram showing the construction of a computer shown in Fig. 1.

[Fig. 3] Illustrative diagram showing the structure of a service agent in the actual example.

[Fig. 4] Illustrative diagram of a connected state of communication path among routine managers in the actual example.

[Fig. 5] Illustrative diagram showing the connected state of a new routine manager with respect to the connected state of Fig. 4.

[Fig. 6] Illustrative diagram showing the connected state of routine managers and service entities in the actual example.

[Fig. 7] Block diagram showing the program structure of a routine manager in the actual example.

[Fig. 8] Illustrative diagram of the structure of tables shown in Fig. 7.

[Fig. 9] Block diagram showing the program structure of a cooperative interface 30 in the actual example.

[Fig. 10] Illustrative diagram of the structure of function table 371 shown in Fig. 9.

[Fig. 11] Illustrative diagrams of description format and description example of a movement destination list 91 of a movement destination list summary 202 shown in Fig. 3.

[Fig. 12] Illustrative diagrams of description method and description example of a movement list 101 of a movement destination list summary 203 shown in fig. 3.

[Fig. 13] Flow chart (1) showing a movement processing procedure of the service agent in the actual example.

[Fig. 14] Flow chart (2) showing a movement processing procedure of the service agent in the actual example.

[Fig. 15] Flow chart (3) showing a movement processing procedure of the service agent in the actual example.

[Fig. 16] Flow chart (4) showing a movement processing procedure of the service agent in the actual example.

[Fig. 17] Flow chart (1) showing processing of a cooperative interface in the actual example.

[Fig. 18] Flow chart (2) showing processing of a cooperative interface in the actual example.

[Fig. 19] Flow chart (3) showing processing of a cooperative interface in the actual example.

[Fig. 20] Flow chart (4) showing processing of a cooperative interface in the actual example.

[Fig. 21] Illustrative diagram of an example of a movement destination determining method based on combinations of data items in the actual example.

[Fig. 22] Block diagram showing a constitutional example of a production line management system applied with the present invention.

[Fig. 23] Illustrative diagram showing a structural example of a service agent in the system of Fig. 22.

[Description of the Symbols]

1] computer

3] application program body

4] service entity

10] routine manager

20] service agent

30] cooperative interface

【図2】

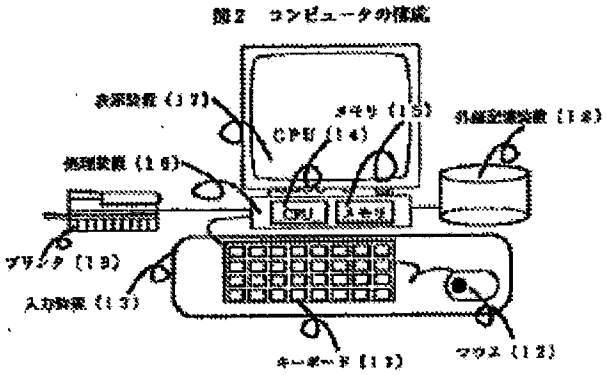


Fig. 2 Construction of computer

11] keyboard

12) mouse

13) input unit

14) CPU

15) memory

16) processing unit

17) display unit

18) external storage unit

19) printer

【図4】

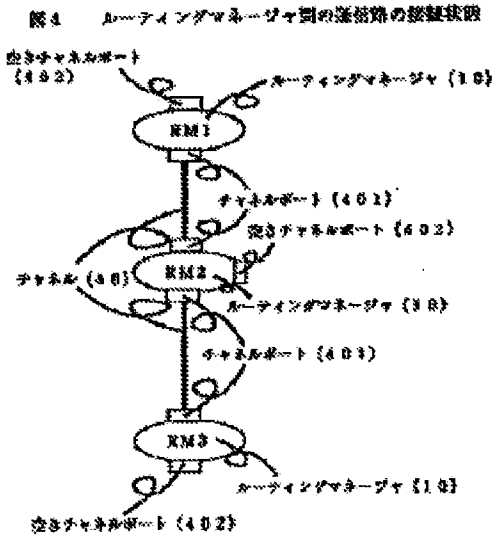


Fig. 4 Connected state of communication path among routine managers

10) routine manager

40] channel

401] channel port

402] vacant channel port

/22

【図1】

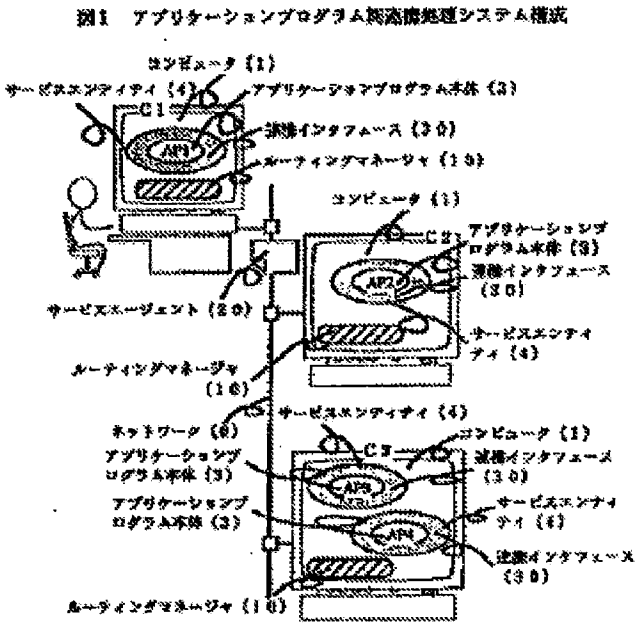


Fig. 1 Constitution of cooperative processing system among application programs

0] network

1] computer

3] application program body

4] service entity

10] routine manager

20] service agent

30] cooperative interface

【図3】

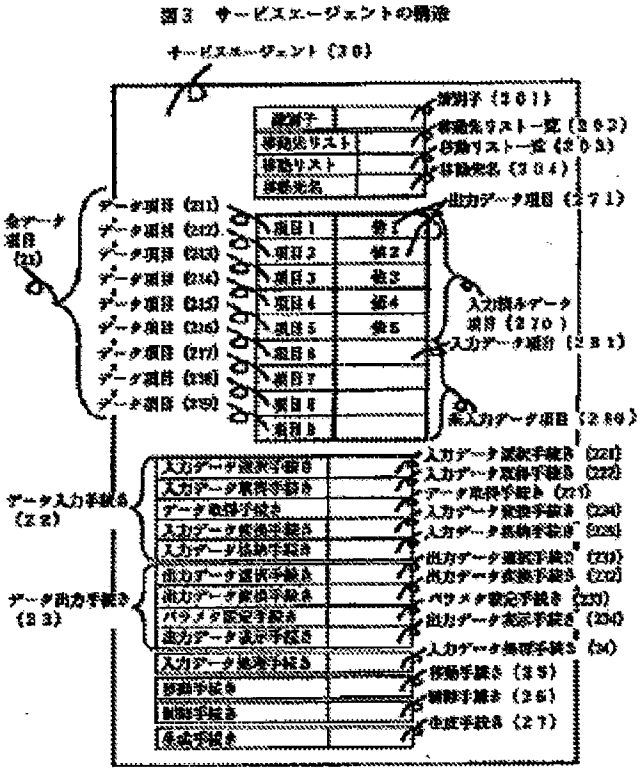


Fig. 3 Structure of service agent

20] service agent

21] all data item

22] data input procedure

23] data output procedure

24] input data processing procedure

25] movement procedure

26] control procedure

27] generation procedure

201] identifier

202] movement destination list summary

203] movement list summary

204] movement destination name

211] data item

212] data item

213] data item

214] data item

215] data item

216] data item

217] data item

218] data item

219 ⌋ data item

221 ⌋ input data selection procedure

222 ⌋ input data acquisition procedure

223 ⌋ data acquisition procedure

224 ⌋ input data conversion procedure

225 ⌋ input data storage procedure

231 ⌋ output data selection procedure

232 ⌋ output data conversion procedure

233 ⌋ parameter setup procedure

234 ⌋ output data display procedure

(table)

Identifier	
Movement destination list	
Movement list	
Movement destination name	

Item 1	Value 1
Item 2	Value 2
Item 3	Value 3
Item 4	Value 4
Item 5	Value 5

Item 6	
Item 7	
Item 8	
Item 9	

Input data selection procedure	
Input data acquisition procedure	
Data acquisition procedure	
Input data conversion procedure	
Input data storage procedure	
Output data selection procedure	
Output data conversion procedure	
Parameter setup procedure	
Output data display procedure	
Input data processing procedure	
Movement procedure	
Control procedure	
Generation procedure	

【図5】

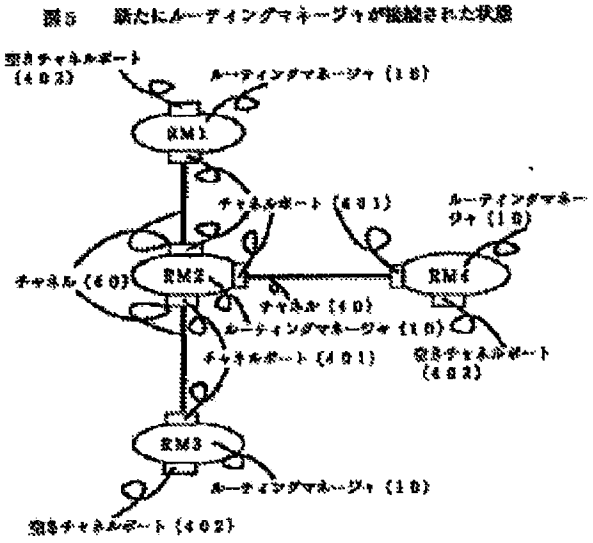


Fig. 5 Connected state of new routine manager

10 ⎫ routine manager

40 ⎫ channel

401 ⎫ channel port

402 ⎫ vacant channel port

【図6】

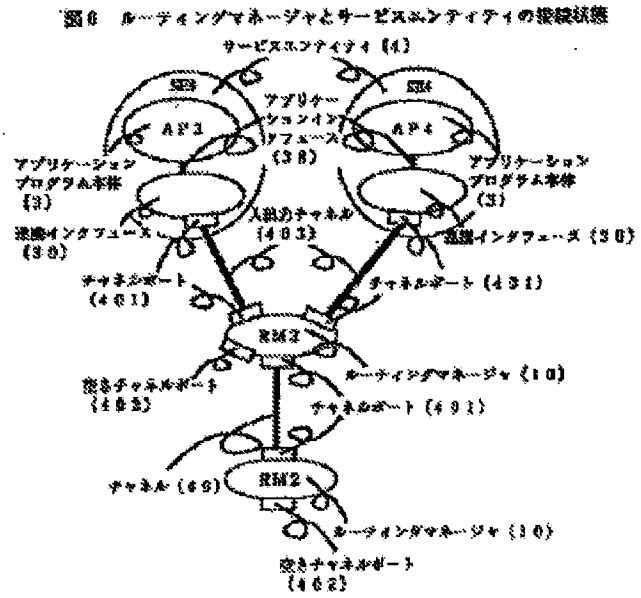


Fig. 6 Connected state of routine managers and service entities

3] application program body

4] service entity

10] routine manager

30] cooperative interface

38] application interface

40] channel

401] channel port

402] vacant channel port

403] input/output channel

/23

【図7】

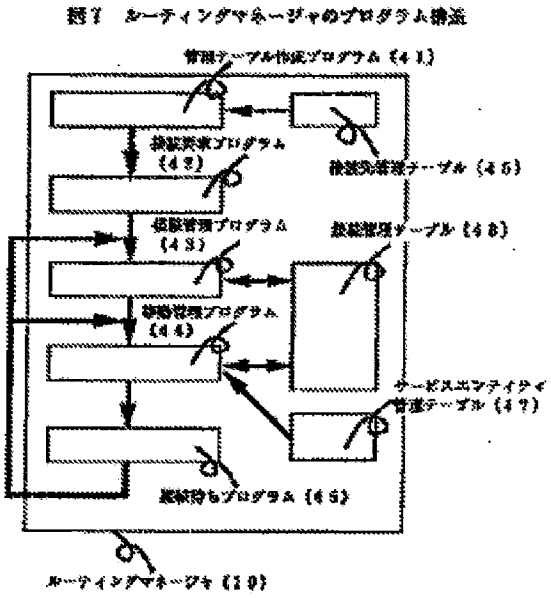


Fig. 7 Program structure of routine manager

10] routine manager

41] management table preparation program

42] connection request program

43] connection management program

44] movement management program

45] connection waiting program

46] connection destination management table

47] service entity management table

48 ⌋ connection management table

【図8】

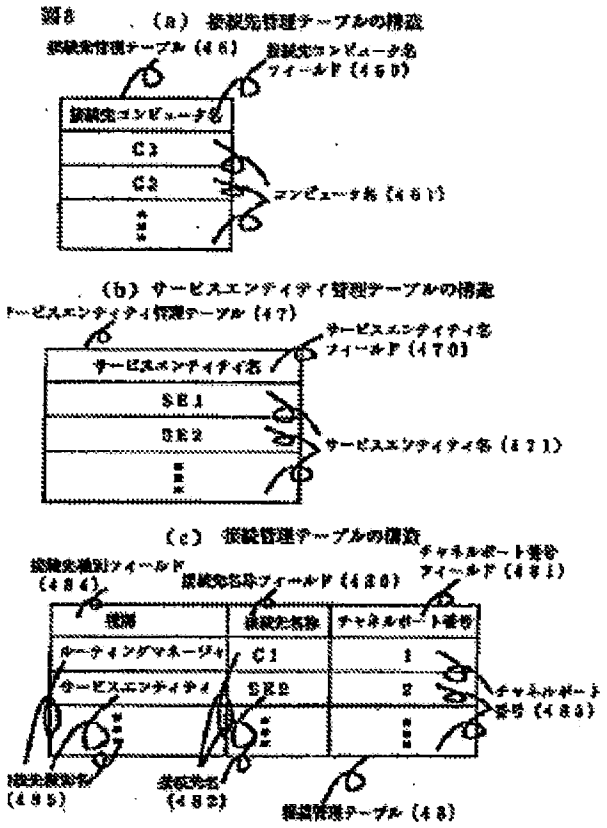


Fig. 8

(a) Structure of connection destination management table 46

460 ⌋ connection destination computer name field

461 ⌋ computer name

(table)

Connection destination computer name

C1
C2
⋮

(b) Structure of service entity management table

470 ⌋ service entity name field

471 ⌋ service entity name

Service entity name
SE1
SE2
⋮

(c) Structure of connection management table

48 ⌋ connection management table

480 ⌋ connection name field

481 ⌋ channel port number field

482 ⌋ connection destination name

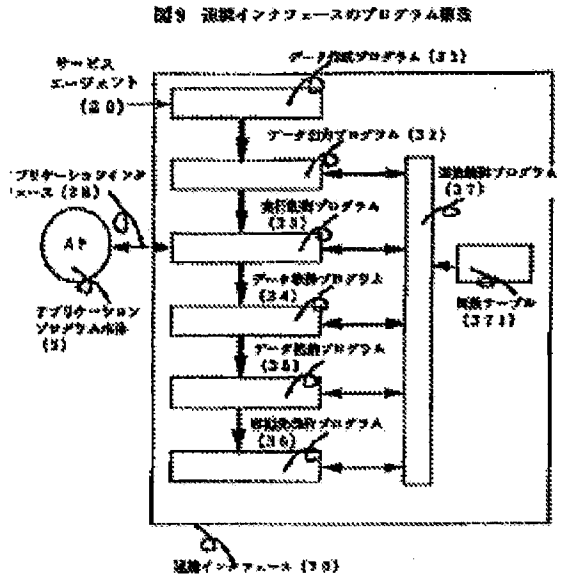
483 ⌋ channel port number

484 ⌋ connection destination kind field

485 ⌋ connection destination kind name

Kind	Connection destination name	Channel port No.
Routine manager	C1	1
Service entity	SE2	2
⋮	⋮	⋮

【図9】



3 ⌋ application program body

20 ⌋ service agent

30 ⌋ cooperative interface

31 ⌋ data preparation program

32 ⌋ data output program

33 ⌋ execution control program

34] data acquisition program

35] data storage program

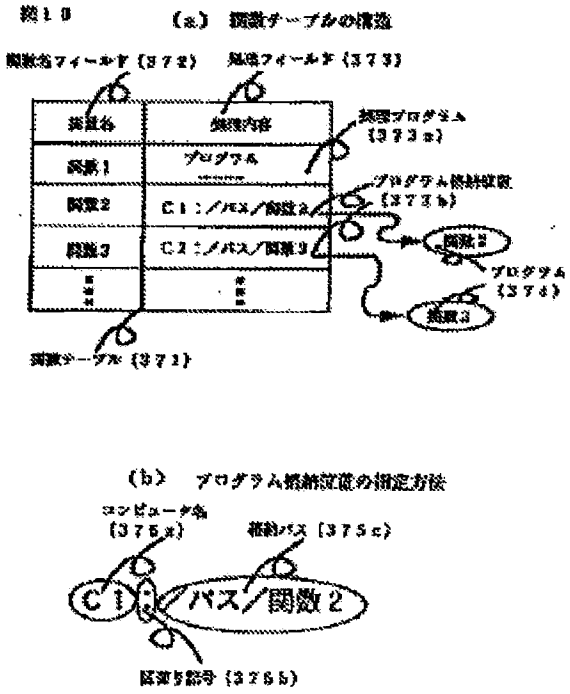
36] movement destination operation program

37] function control program

371] function table

38] application interface

【図１０】



(a)

(b) Structure of function table

371] function table

372] function name field

373 ⌋ processing field

373a ⌋ processing program

373b ⌋ program storage position

374 ⌋ program function 2 function 3

(table)

Function name	Processing contents
Function 1	Program ⌋⌋⌋⌋
Function 2	C1:/bus/function 2
Function 3	C2:/bus/function 3
⋮	⋮

(b) Method for assigning program storage position

C1:/pass/function 2

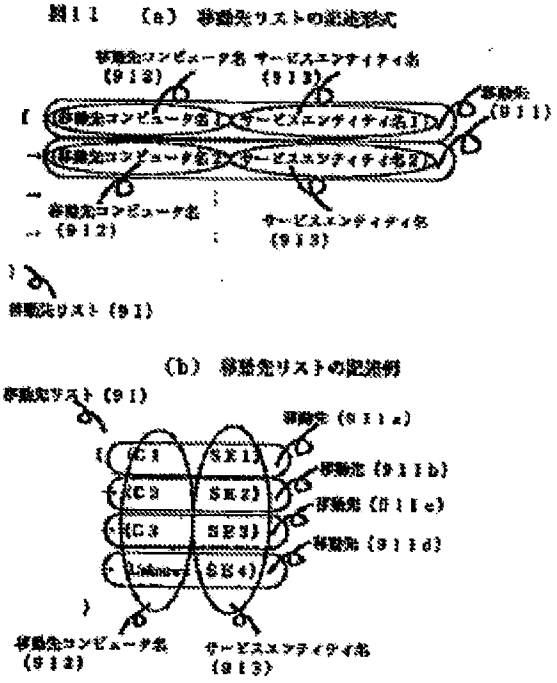
375a ⌋ computer name

375b ⌋ breakoff symbol

375c ⌋ storage pass

/24

【図１１】



(a) Description format of movement destination list

91 ⌋ movement destination list

911 ⌋ movement destination

912 ⌋ movement destination computer name

movement destination computer name 1

movement destination computer name 2

913 ⌋ service entity name

service entity name 1

service entity name 2

(b) Description example of movement destination list

91 ⎫ movement destination list

911a ⎫ movement destination

911b ⎫ movement destination

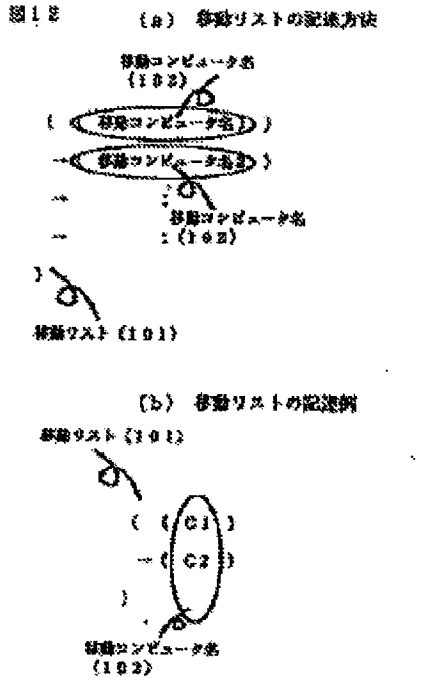
911c ⎫ movement destination

911d ⎫ movement destination

912 ⎫ movement destination computer name

913 ⎫ service entity name

【図12】



(a) Description format of movement list

101 ⌋ movement list

102 ⌋ movement computer name

movement computer name 1

movement computer name 2

(b) Description example of movement list

101 ⌋ movement list

102 ⌋ movement computer name

【図21】

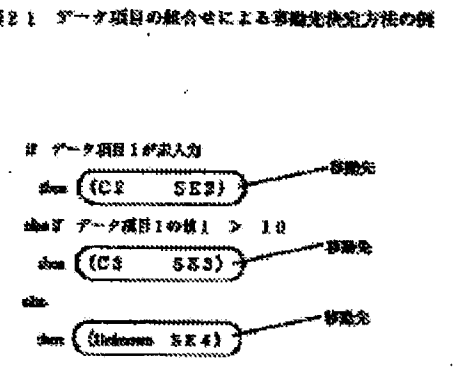


Fig. 21 Example of movement destination determining method based on combinations of data items

If data items are not inputted,

then (C2 SE2) ——— movement destination

else if the value of data item 1 > 10

then (C3 SE3) ———— movement destination

else

then (Unknown SE4) ———— movement destination

【図22】

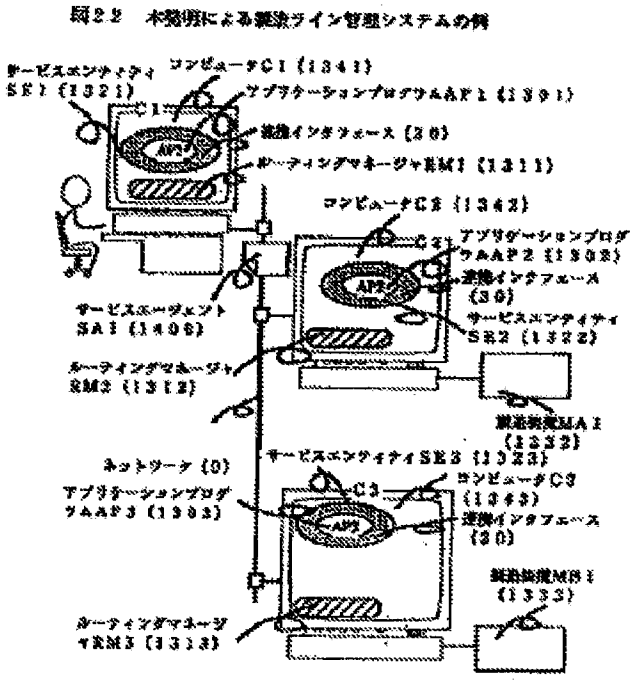


Fig. 22 Example of production line management system based on present invention

0 ⎫ network

30 ⎭ cooperative interface

1301 ⎫ application program AP1

1302 ⌡ application program AP2

1303 ⌡ application program AP3

1311 ⌡ routine manager RM1

1312 ⌡ routine manager RM2

1313 ⌡ routine manager RM3

1321 ⌡ service entity SE1

1322 ⌡ service entity SE2

1323 ⌡ service entity SE3

1332 ⌡ production machine MA1

1333 ⌡ production machine MB1

1341 ⌡ computer C1

1342 ⌡ computer C2

1343 ⌡ computer C3

1400 ⌡ service agent SA1

/25

【図13】

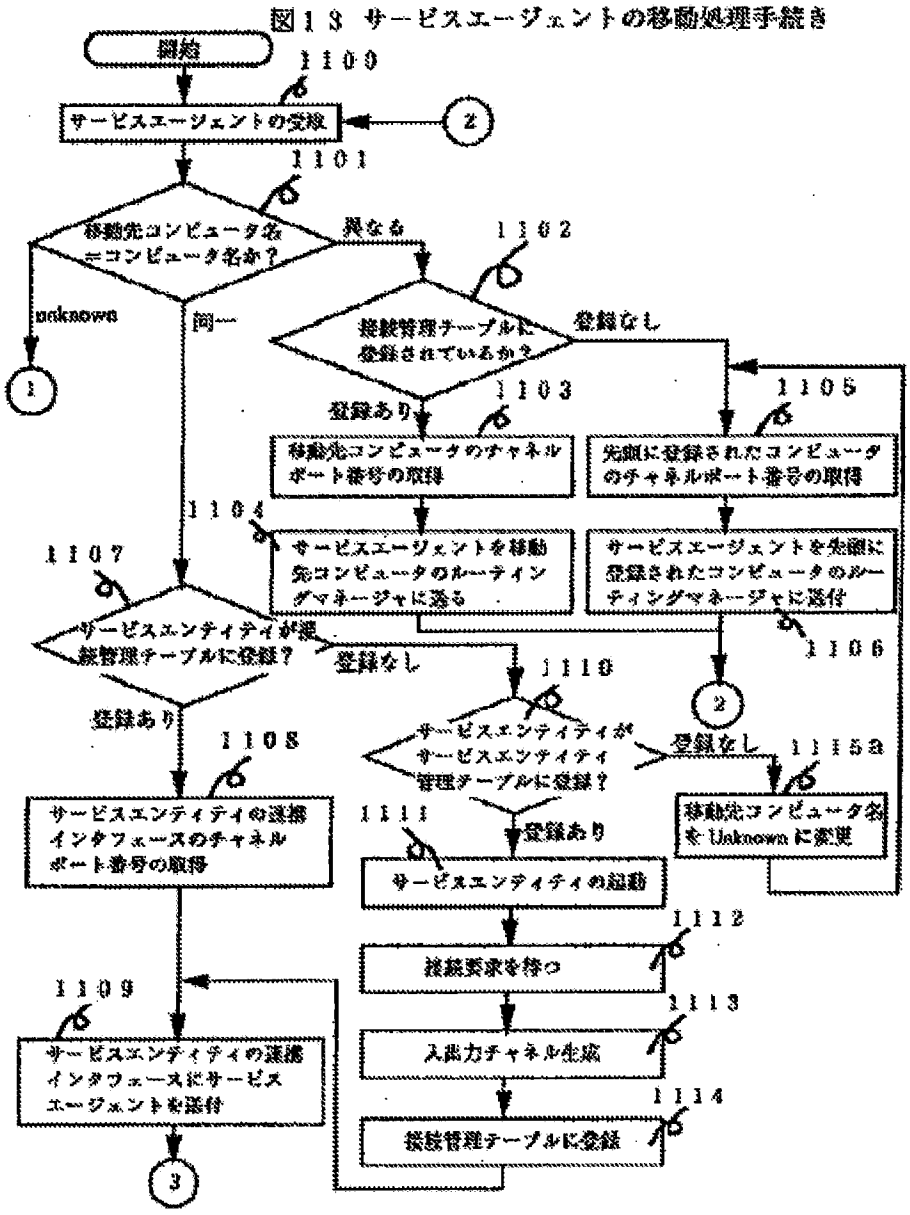


Fig. 13 Movement processing procedure of service agent

START

1100 ⌋ Receive service agent

1101 ⌋ destination computer name = computer name?

1102 ⌋ Is registered in connection management table?

1103 ⌋ Acquire channel port number of movement destination computer

1104 ⌋ Send service agent to routine manager of movement destination computer

1105 ⌋ Acquire channel port number of computer registered in head

1106 ⌋ Send service agent to routine manager of computer registered in head

1107 ⌋ Is service entity registered in connection management table?

1108 ⌋ Acquire channel port number of cooperative interface of service entity

1109 ⌋ Send service agent to cooperative interface of service entity

1110 ⌋ Is service entity registered in service entity management table?

1111 ⌋ Start service entity

1112 ⌋ Wait for connection request

1113 ⌋ Generate input/output channel

1114 ⌋ Register in connection management table

1115a ⌋ Change movement destination computer name to Unknown

(between 1101 and 1102) No

(between 1101 and 1107) Yes

(between 1102 and 1103) Yes

(between 1102 and 1105) No

(between 1107 and 1108) Yes

(between 1107 and 1110) No

(between 1110 and 1111) Yes

(between 1110 and 1115a) No

/26

【図14】

図14 サービスエージェントの移動処理手続き

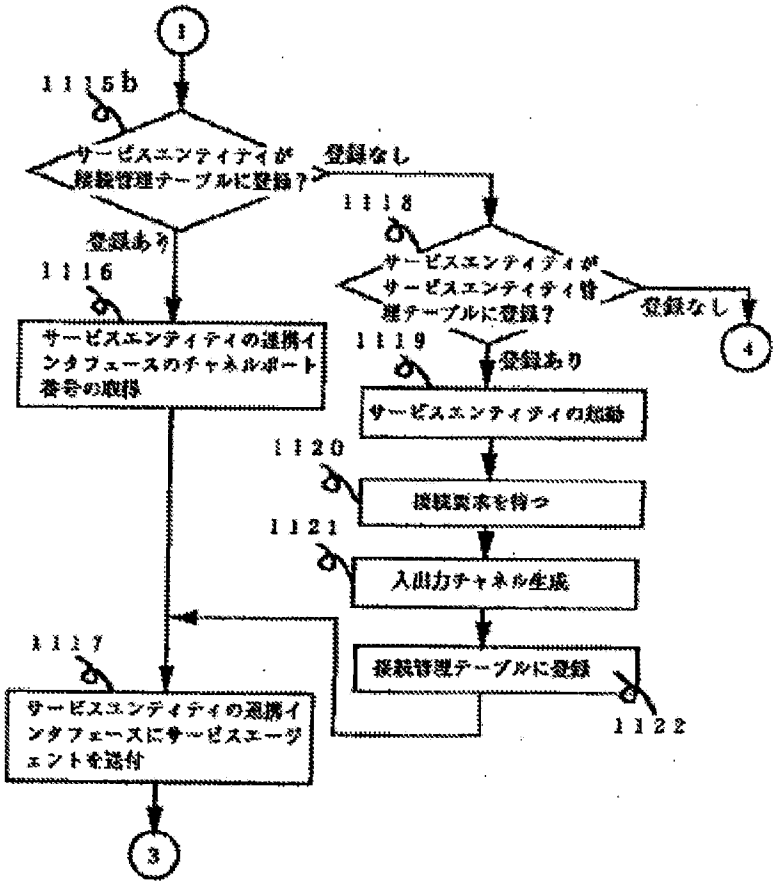


Fig. 14 Movement processing procedure of service agent

1115b] Is service entity registered in connection management table?

1116] Acquire channel port number of cooperative interface of service entity

1117 ⌋ Send service agent to cooperative interface of service entity

1118 ⌋ Is service entity registered in service entity management table?

1119 ⌋ Start service entity

1120 ⌋ Wait for connection request

1121 ⌋ Generate input/output channel

1122 ⌋ Register in connection management table

(between 1115b and 1116) Yes

(between 1115b and 1118) No

(between 1118 and 1119) Yes

(between 1118 and 4) No

/27

[図15]

図15 サービスエージェントの移動処理手続き

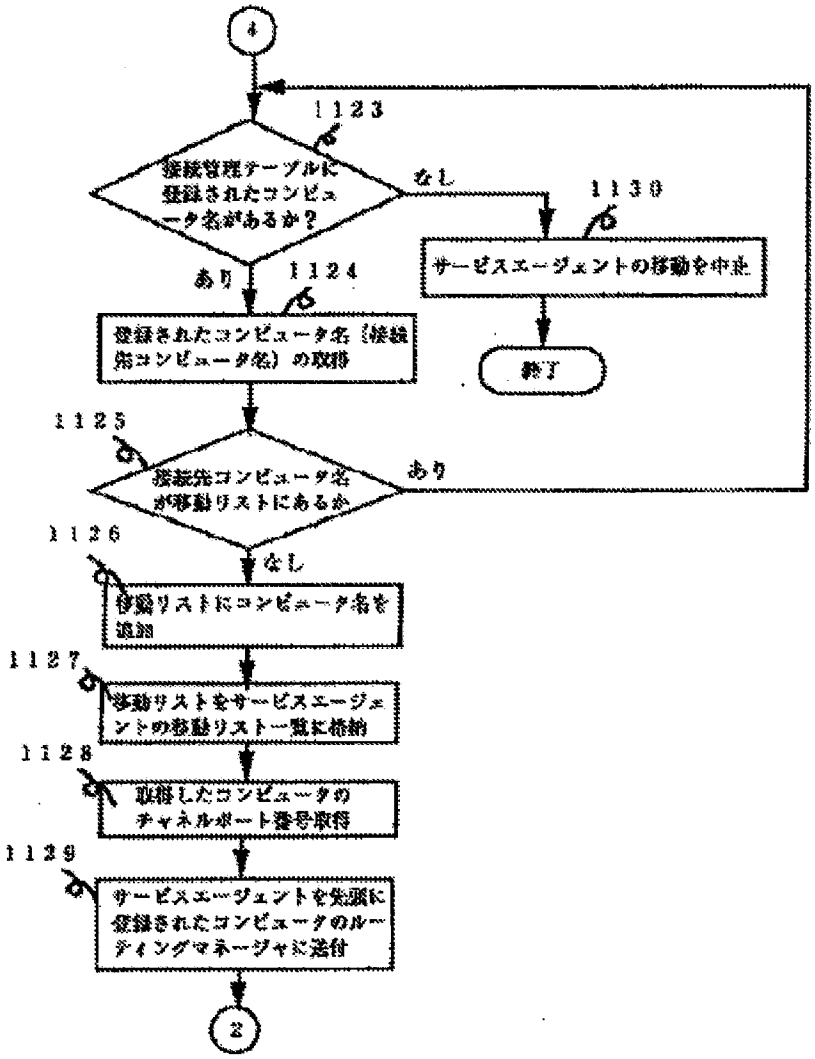


Fig. 15 Movement processing procedure of service agent

1123) Does registered computer name exist in connection management table?

1124) Acquire registered computer name (connection desti-

nation computer name)

1125 ⌋ Is connection destination computer name in movement list?

1126 ⌋ Supplement computer name in movement list

1127 ⌋ Store movement list in movement list summary of service agent

1128 ⌋ Acquire channel port number of acquired computer

1129 ⌋ Send service agent routine manager of computer registered in head

1130 ⌋ Stop movement of service agent

End

(between 1123 and 1124) Yes

(between 1123 and 1130) No

(between 1125 and 1123) Yes

(between 1125 and 1126) No

/28

【図16】

図16　サービスエージェントの移動処理手続き

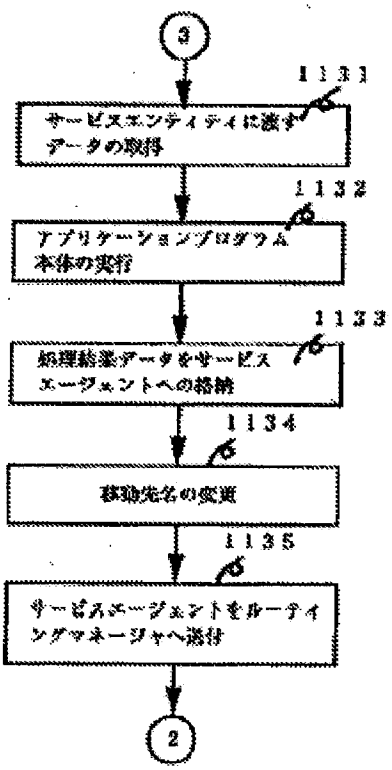


Fig. 16　Movement processing procedure of service agent

1131 ⎫　Acquire data transferred to service entity

1132 ⎫　Execute application program body

1133 ⎫　Store processing result data into service agent

1131 ⎫　Change movement destination name

1135 ⎫　Send service agent to routine manager

【図23】

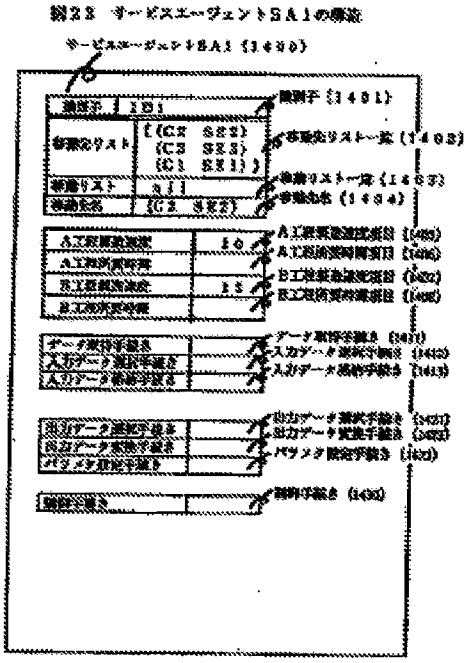


Fig. 23 Structure of service agent SA1

1400 ⌋ service agent SA1

1401 ⌋ identifier

1402 ⌋ movement destination list summary

1403 ⌋ movement list summary

1404 ⌋ movement destination name

1405 ⌋ process A manufacturing speed item

1406 ⌋ process A required time item

1407 ⌋ process B manufacturing speed item

1408 ⌋ process B required time item

1411 ⌋ data acquisition procedure

1412 ⌋ input data selection procedure

1413 ⌋ input data storage procedure

1421 ⌋ output data selection procedure

1422 ⌋ output data conversion procedure

1423 ⌋ parameter setup procedure

1430 ⌋ control procedure

(table)

Identifier	ID1
Movement destination list	((C2 SE2) (C3 SE3) (C1 SE1))
Movement list	nil
Movement destination name	(C2 SE2)

Process A manufacturing speed	10
Process A required time	
Process B manufacturing speed	15
Process B required time	

Data acquisition procedure	
Input data selection procedure	
Input data storage procedure	

Output data selection procedure	
Output data conversion procedure	
Parameter setup procedure	

Control procedure	

/29

【図17】

図17 連携インタフェースの処理フロー

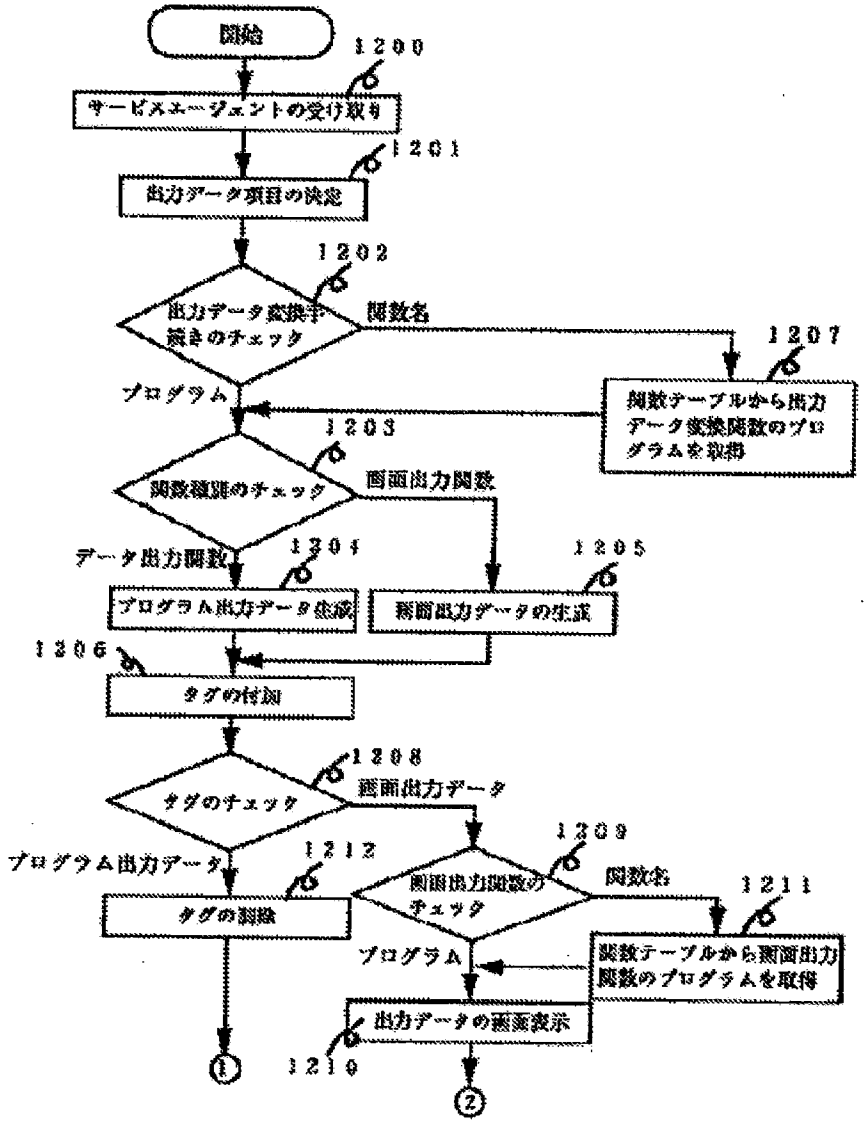


Fig. 17 Processing flow of cooperative interface

Start

1200 ⌋ Receive service agent

1201 ⌋ Determine output data item

1202 ⌋ Check output data conversion procedure

1203 ⌋ Check kind of function

1204 ⌋ Generate program output data

1205 ⌋ Generate picture output data

1206 ⌋ Attach tag

1207 ⌋ Acquire program of output data conversion function from function table

1208 ⌋ Check tag

1209 ⌋ Check picture output function

1210 ⌋ Display picture of output data

1211 ⌋ Acquire program of output data function from function table

(between 1202 and 1203) Program

(between 1202 and 1207) Function name

(between 1203 and 1204) Data output function

(between 1203 and 1205) Picture output function

(between 1208 and 1209) Picture output data

(between 1208 and 1212) Program output data

(between 1209 and 1210) Program

(between 1209 and 1211) Function name

/30

【図18】

図18　連携インタフェースの処理フロー

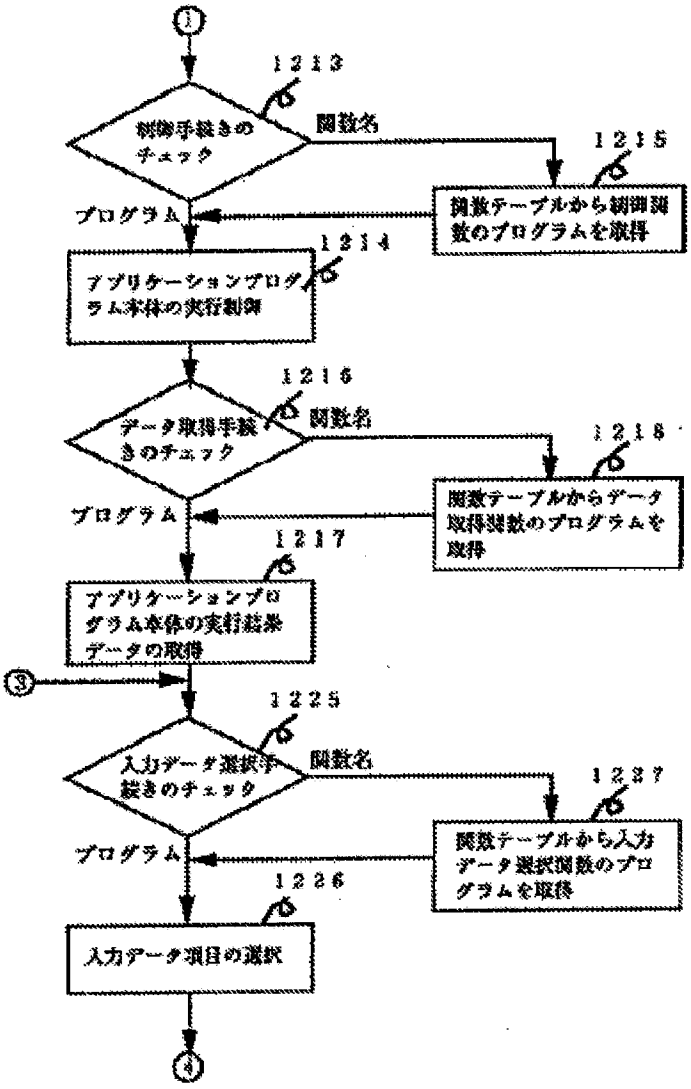


Fig. 18　Processing flow of cooperative interface

1213 ⎫　Check control procedure

1214 ⎭　Control execution of application program body

1215 ⌋ Acquire program of control function from function table

1216 ⌋ Check data acquisition procedure

1217 ⌋ Acquire execution result data of application program body

1218 ⌋ Acquire program of data acquisition function from function table

1225 ⌋ Check input data selection procedure

1226 ⌋ Select input data item

1227 ⌋ Acquire program of input data selection function from function table

(between 1213 and 1214) Program

(between 1213 and 1215) Function name

(between 1225 and 1226) Program

(between 1225 and 1227) Function name

/31

【図19】

図19　連携インタフェースの処理フロー

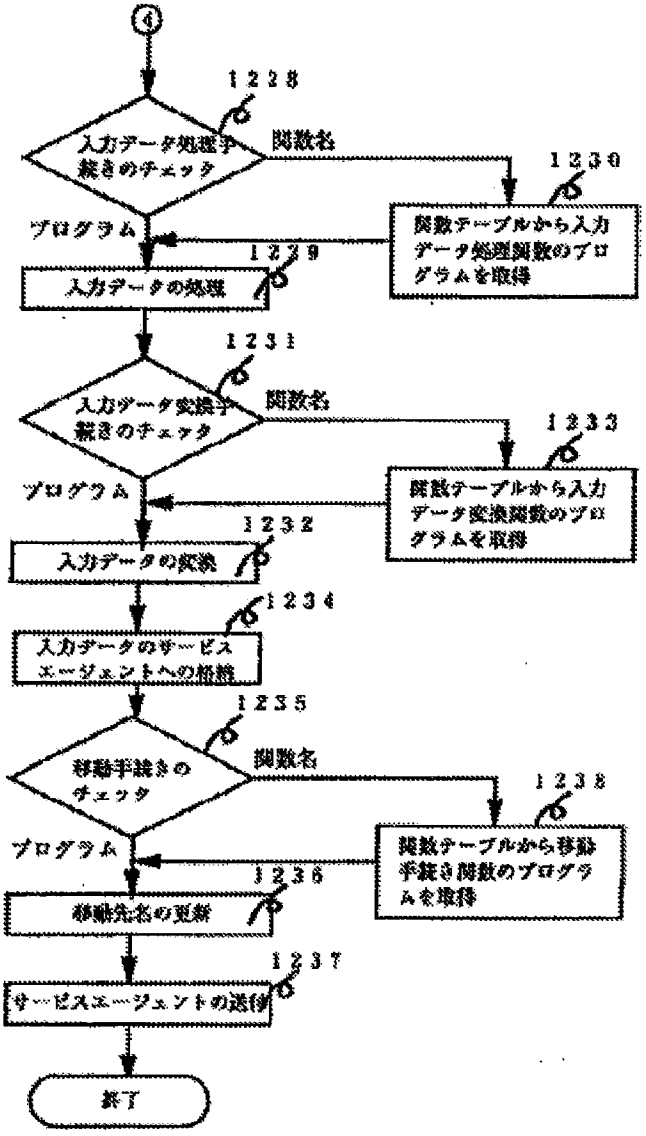


Fig. 19　Processing flow of cooperative interface

1228 ⌋　Check input data processing procedure

1229 ⌋ Processing of input data

1230 ⌋ Acquire program of input data processing function from function table

1231 ⌋ Check input data conversion procedure

1232 ⌋ Convert input data

1233 ⌋ Acquire program of input data conversion function from function table

1234 ⌋ Store input data into service agent

1235 ⌋ Check movement procedure

1236 ⌋ Renew movement destination name

1237 ⌋ Send service agent

1238⌋ Acquire program of movement procedure from function table

End

(between 1228 and 1229) Program

(between 1228 and 1229) Function name

(between 1231 and 1232) Program

(between 1231 and 1233) Function name

(between 1235 and 1236) Program

(between 1235 and 1238) Function name

[図20]

図２０　連携インタフェースの処理フロー

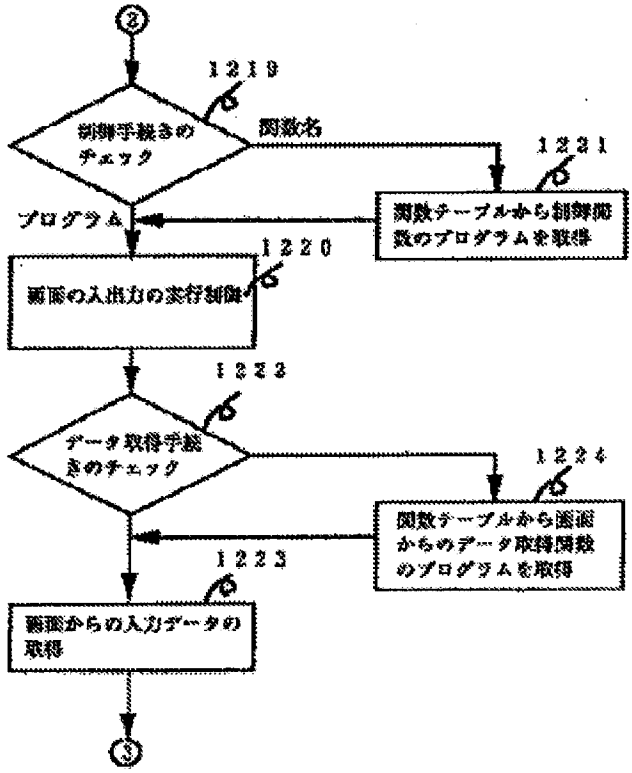


Fig. 20　Processing flow of cooperative interface

1219 ｝　Check control procedure

1220 ｝　Control execution of picture input/output

1221 ｝　Acquire program of control function from function table

1222 ｝　Check data acquisition procedure

1223 ｝　Acquire input data from picture

1224 ⌋ Acquire program of data acquisition from picture from function table

(between 1219 and 1220) Program

(between 1219 and 1221) Function name